APLICACIÓN DEL APRENDIZAJE PROFUNDO (“*DEEP LEARNING”*) AL PROCESAMIENTO DE SEÑALES DIGITALES

GLEN JHAN PIERRE RESTREPO ARTEAGA

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2015

APLICACIÓN DEL APRENDIZAJE PROFUNDO (*DEEP LEARNING*) AL PROCESAMIENTO DE SEÑALES DIGITALES

GLEN JHAN PIERRE RESTREPO ARTEAGA

Proyecto de grado para optar al título de Ingeniero Mecatrónico

Director
JESÚS ALFONSO LÓPEZ SOTELO
Ingeniero Electricista
Magíster en Automática
Doctor en Ingeniería

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA DE INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2015

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Mecatrónico

JIMMY TOMBÉ ANDRADE
Jurado

JUAN CARLOS MENA MORENO
Jurado

Santiago de Cali, Marzo 26 de 2015

CONTENIDO

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

RESUMEN

Este trabajo hace una revisión del concepto de *deep learning*, en la cual se muestran las diferentes arquitecturas que cumplen con dicho concepto y como estas se relacionan con las redes neuronales artificiales (RNA), tanto por su arquitectura, como por su tipo de entrenamiento. Realizando de esta forma, una taxonomía que explica o clasifica algunas de las redes que conforman el *deep learning*.

Con la revisión de arquitecturas y aplicaciones, se elaboró una aplicación en MATLAB con dos de las arquitecturas más comunes del concepto, en la cual se puede apreciar el comportamiento distintivo de estas arquitecturas, con lo que se corrobora la aplicabilidad del *deep learning* en el procesamiento digital de imágenes, obteniendo buenos resultados de clasificación.

PALABRAS CLAVE: *Deep learning, deep belief network, convolutional neural network, stacked denoising autoencoder, deep Boltzmann machine, convolutional deep belief network, deep stacking network, multilayer kernel machine, spike-and-slab RMBs, deep predictive coding network,* aprendizaje supervisado, aprendizaje no supervisado, procesamiento digital de imágenes, MATLAB.

INTRODUCCIÓN

Las redes neuronales artificiales (RNA) se desarrollaron inicialmente a lo largo de la década de los años 50 con la presentación a la comunidad científica del Perceptrón y la red ADALINE. La primera aprende a reconocer patrones sencillos con una superficie de separación lineal; mientras que la segunda, en vez de tener una función de activación binaria, tiene una función de activación lineal, además de basarse en el uso del gradiente descendente para su regla de entrenamiento, la cual hoy en día, es utilizada en la mayoría de algoritmos de aprendizaje.

Después de las fuertes críticas que recibieron las dos anteriores RNA y del abandono, como consecuencia de lo anterior, fueron un gran adelanto para este campo. Años más tarde, se creó el algoritmo más conocido, denominado *backpropagation*, el cual se basa de varias capas de procesamiento antes de la capa de salida, para separar patrones que son linealmente no separables; lo que permitiría el desarrollo de mas RNA mucho más avanzadas.

Actualmente, el desarrollo de las RNA se ha implicado más en la parte práctica que en la parte biológica, creando RNA con capacidades de procesamiento similar a la de un humano y cada vez más cerca de llegar a simular sistemas tan complejos como el córtex visual.

Existen múltiples aplicaciones de las RNA, como organización o clasificación de datos, detectores de patrones, reconocimiento de voz, etc. Sin embargo, cada vez las aplicaciones hacen que los clásicos algoritmos o modelos se vayan volviendo ineficientes; por lo que se ha desarrollado un algoritmo denominado *"deep learning"* el cual se ido incorporando en el ámbito de las RNA. Esta técnica se considera un poco emergente debido a que la misma no es muy reconocida, sin embargo, compañías como Facebook y Google hacen uso de esta para aumentar los atributos en sus aplicaciones web entre otros. Por lo anterior, se verificará la aplicabilidad del aprendizaje profundo "*deep learning"* en el procesamiento digital de imágenes usando como plataforma de desarrollo MATLAB.

1. PLANTEAMIENTO DEL PROBLEMA

El gran problema de "construir" un sistema capaz de funcionar como una red neuronal biológica ha sido la inspiración de investigadores[1], pero a medida que se desarrollan nuevos algoritmos se topan con distintos conflictos y/o falta de recursos, computacionalmente hablando[2]. Los primeros modelos de RNA fueron el perceptrón y el ADALINE, que poseen la capacidad de separar patrones linealmente separables, limitándolos solo a este tipo de problemas, por lo que poco a poco se perdió el interés por los anteriores.

Los primeros modelos solo poseían una capa de procesamiento, la de salida, por lo que se desarrollaron algoritmos que tuvieran capas "ocultas" entre la capa de entrada y la de salida que también se encargaran del procesamiento y clasificación de los patrones de entrada, desarrollando el perceptrón multicapa o MLP. El anterior se basa del algoritmo de entrenamiento "*backpropagation"*, el cual se encarga de propagar los errores de las salidas hacia las entradas (de ahí su nombre). A pesar de solucionar el problema del perceptrón simple, el MLP también posee limitaciones, como depender del entrenamiento, es decir, si no se entrena la red lo suficiente, la red puede dar salidas imprecisas, además se el algoritmo de entrenamiento (*backpropagation"*) puede verse afectado por un mínimo local de la función de error, no llegando al mínimo global, el cual es el deseado.

De esta manera las RNA desarrolladas presentan numerosos problemas que, entre los más comunes son: Las limitaciones de hardware para la implementación de una RNA de múltiples capas, la definición de muchos parámetros antes de entrenar a la red, como lo son: el número de capas ocultas, la cantidad de iteraciones, la función de transferencia, el algoritmo de entrenamiento, etc. Además, si se requiere adicionar un nuevo parámetro o conocimiento, se necesita cambiar las interacciones entre las capas para que el efecto de este, se unificado y se sintetice el nuevo parámetro[3].

---

[1] Artificial intelligence scientist gest $1M prize [en línea]. CBC News, 2011. [Consultado 25 de marzo, 2014]. Disponible en Internet: http://www.cbc.ca/news/technology/artificial-intelligence-scientist-gets-1m-prize-1.1034093#ixzz1DxUEvAcQ

[2] CLAUDIU CIRESAN, Dan, et al. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition [en línea]. arXiv preprint arXiv:1003.0358, 2010. p. 1. [Consultado 01 de abril, 2014]. Disponible en Internet: http://arxiv.org/pdf/1003.0358.pdf

[3] TORRES SOLER, Luis Carlos. Redes Neuronales [en línea]. Diciembre 2010. [Consultado 11 de abril, 2014]. Disponible en Internet: http://disi.unal.edu.co/~lctorress/RedNeu/LiRna008.pdf

Recientemente, se ha desarrollado un nuevo concepto llamado aprendizaje profundo ("*deep learning*"), el cual no necesita de muchos parámetros, pues el mismo se encarga de realizar una jerarquía de aprendizaje el cual es más eficiente que las anteriores RNA concebidas, pero ¿Es posible aplicar el aprendizaje profundo en el procesamiento digital de señales?

2. JUSTIFICACIÓN

Con el claro avance de la tecnología en los campos de la información, desarrollo de productos, comunicación y software, el enfoque de las industrias, empresas y universidades por obtener nuevas tecnologías que permitan innovar ha brindado la oportunidad de que nuevas técnicas puedan aplicarse en beneficio en diferentes sectores para la generación de nuevas aplicaciones, y este es el caso del aprendizaje profundo "*deep learning"*.

Actualmente la Universidad Autónoma de Occidente cuenta con un grupo de investigación llamado "GITCOD" el cual posee una línea de énfasis en sistemas inteligentes, lo que hace que este proyecto deje bases significativas para el desarrollo de nuevas aplicaciones en esta línea, además de tener herramientas que se puedan aplicar en diferentes investigaciones realizadas en la universidad y trabajar con una técnica novedosa en nuestro entorno que no se ha abordado antes.

El presente proyecto se enfoca en el estudio y análisis de los diferentes modelos de RNA que se han desarrollado en la actualidad, especialmente en una técnica denominada aprendizaje profundo (*"deep learning"*), ya que estas han jugado y juegan un papel importante a la hora de desarrollar determinadas aplicaciones y aprovechar las capacidades que brinda cada modelo. El "deep learning" es una técnica relativamente nueva, pues se sigue investigando y desarrollando desde el 2006, año en que fue retomada la investigación[4], por parte de renombrados investigadores en el campo de inteligencia artificial y RNA que son contratados por diferentes compañías para la aplicación de dicha técnica.

Por lo anterior, se pretende desarrollar una aplicación en donde se evidencie las características que distinguen a este nuevo concepto con respecto a sus predecesores, buscando problemas en el campo procesamiento digital de imágenes que ameriten su uso.

[4] GULCEHRE, Caglar. Deep Learning – Bibliography [en línea]. Septiembre 2014. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://deeplearning.net/bibliography/

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Ilustrar la aplicabilidad del aprendizaje profundo ("*deep learning*") en el procesamiento digital de señales unidimensionales e imágenes.

3.2. OBJETIVOS ESPECÍFICOS

- Estudiar el funcionamiento de la técnica de aprendizaje profundo (*"deep learning"*) contrastándolo con el funcionamiento de las redes neuronales por medio de sus respectivas aplicaciones.

- Seleccionar aplicaciones del aprendizaje profundo en procesamiento digital de imágenes.

- Elaborar una herramienta computacional en MATLAB donde se muestren ejemplos de una aplicación del aprendizaje profundo ("*deep learning*") al procesamiento digital de imágenes.

4. CONCEPTO DE DEEP LEARNING

4.1. ANTECEDENTES

Las investigaciones cada vez exhaustivas acerca del funcionamiento del celebro y del paradigma de la inteligencia humana han permitido que la inteligencia artificial evolucione, por medio de las RNA, para elaborar un sistema semejante capaz de lograr una semejante hazaña. Desde 1943, en los primeros intentos por elaborar una red neuronal[5], se han construyendo modelos más sofisticados o robustos, con un nivel de acercamiento cada vez mayor al comportamiento del celebro, hasta llegar al punto del aprendizaje profundo o *deep learning*.

A principios de los años 80, se desarrolló un concepto que, más tarde (2006) se conocería como *deep learning* por medio del investigador Geoffrey Hinton, que ayudo a lanzar una revolución ambiciosa al explorar las RNA, en la universidad de Cambridge y la Universidad de Edimburgo, la cual consistía en imitar el celebro por medio del uso de hardware y software, para crear una forma más pura de inteligencia artificial[6]. Sin embargo, Hinton y sus colegas al trabajar con la mencionada idea, las computadoras no tenían la capacidad computacional requerida para procesar las enormes colecciones de datos que requerían las RNA, por lo que su éxito fue limitado; con lo que la comunidad de AI les dio la espalda.

En 1991, el investigador Jürgen Schmidhuber elabora un red de cientos de operadores lineales o capas de neuronas, el cual, mediante el uso de un "pre-entrenamiento no supervisado", entrena otra red neuronal recurrente (RNN); generando una memoria temporal jerárquica, que comprime los datos. Esta, según su mismo actor, es una de las primeras técnicas de *deep learning* en superar el problema asociado con el colapso parcial de una pérdida de memoria jerárquica, debido a la compresión de los niveles más altos hacia los bajos de las capas de "pre-entrenamiento".

Tiempo después (1997), Jürgen y su colega Sepp Hochreiter desarrollaron una RNN denominada "Long short term memory (LSTM)", con la cual, además se

[5] LÓPEZ S, Jesús A. CAICEDO B, Eduardo F. "Una aproximación práctica a las redes neuronales artificiales". Editorial Universidad Del Valle, 2009, p.3.
[6] HERNANDEZ, Daniela. Meet the Man Google Hired to Make AI a Reality [en línea]. Enero 2014. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://www.wired.com/wiredenterprise/2014/01/geoffrey-hinton-deep-learning

resolver el problema del colapso, evitaban el "pre-entrenamiento no supervisado", mejorando el algoritmo de 1991[7].

En 2004, con algo una pequeña cantidad de fondos otorgada por el Instituto Canadiense para la Investigación Avanzada (CIFAR, por sus siglas en inglés) y el apoyo de Yann LeCun y Yoshua Bengio, Hinton fundó el programa "Neural Computation and Adaptive Perception", un grupo selecto de científicos de la computación, biólogos, ingenieros eléctricos, neurocientíficos, físicos y psicólogos; esto permitió la búsqueda de pensadores o investigadores de talla mundial con los cuales, además de permitir el avance del concepto de *deep learning* (iniciado por Geoffrey Hinton), se dedicarían a la creación de sistemas informáticos que imiten la inteligencia ecológica.

Con la creación publicación de artículos y conferencias sobre esta tecnología, en 2006, se le acuño el término de *deep learning* en el contexto del "pre-entrenamiento no supervisado" para redes feedforward que tenían una jerarquía de memoria. Tres años más tarde (2009), el grupo de Jürgen gano una competencia internacional con una RNN LSTM, convirtiéndose en la primer RNN y sistemas *deep learning* en ganar una competencia de esta talla en el reconocimiento de patrones junto con varios concursos de escritura a mano conectados a ICDAR 2009, los cuales también los gano la RNN LSTM. Un tiempo después, Alex Graves, un desarrollador del sistema RNN mencionado, se trasladó al laboratorio de Geoffrey, en la Universidad de Toronto.

En los próximos tres años (2012), después de ganar la ICDAR 2009, el grupo de Jürgen siguió trabajando en los sistemas *deep learning*; ahora en las GPUs, mostrando excelentes rendimientos y rompiendo records mundiales en tasas de error de hasta 0,37%, lo que obviamente lo llevaría a reconocimientos y a ganar grandes premios con una técnica denominada "deep MCMPCNN" en 2010, para, un año más tarde desarrollar el primer reconocimiento de patrones visuales sobrehumano, sobrepasando, como su nombre lo indica, el reconocimiento en este campo del ser humano, y, obviamente, de los sistemas RNA anteriores. A esta técnica se le llamo "GPU-MPCNN", pues se basó en una GPU.

Por lo anterior, la "GPU-MPCNN" fue adoptada por los grupos de la Universidad de Toronto, Stanford, Google y Apple Inc.

[7] SCHMIDHUBER Jürgen. "My First Deep Learning System of 1991 + Deep Learning Timeline 1962-2013" [en línea]. Diciembre 2013. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://www.idsia.ch/~juergen/firstdeeplearner.html

Actualmente la inteligencia artificial está de moda, pues parte de los integrantes de Jürgen son contratados por diferentes entidades para el desarrollo de esta novedosa tecnología y por parte del grupo de Geoffrey, el mismo trabaja a tiempo parcial para Google y LeCun está trabajando en Facebook, ambos desde el 2013. Esta novedad también está ingresando en Microsoft, IBM, Baibu, etc.

4.2. CONCEPTOS DE RNA

El *deep learning* es un concepto muy amplio, lo que conlleva a que no tenga solo una definición veraz[8]. Sin embargo, se puede generalizar en que el *deep learning* es un concepto que surge de la idea de imitar el celebro a partir del uso de hardware y software, para crear una inteligencia artificial pura[9], utilizando una capacidad de abstracción jerárquica, es decir, una representación de los datos de entrada en varios "niveles", en el caso de las RNA, en varias capas, para seleccionar características que son útiles para el aprendizaje; de esta manera, una característica de un nivel de complejidad más alto será aprendido de una de un nivel de complejidad más bajo[10].

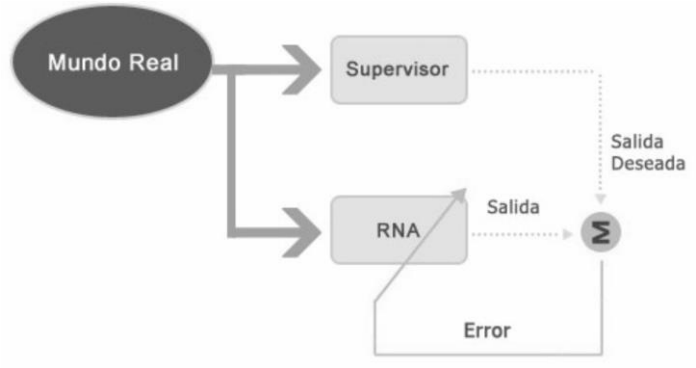
Consecuente con lo anterior, se ha desarrollado un rápido crecimiento en la cantidad de arquitecturas y/o algoritmos de entrenamiento en una RNA, incluso, variantes de ellos, las cuales siguen el concepto planteado anteriormente. Así, las arquitecturas y/o algoritmos desarrollados se basan en otras arquitecturas "más simples"; modificándolas y obteniendo una arquitectura "más profunda", lo que quiere decir que en sus capas y neuronas se separan las características de un conjunto de datos de entrada de una forma jerarquizada.

Dependiendo de la RNA, el algoritmo de entrenamiento de las RNA "más simples", de las cuales están compuestas las arquitecturas profundas, se pueden caracterizar, principalmente, en dos categorías[11]:

- **Supervisado:** Se caracteriza porque su entrenamiento es controlado por un agente externo. Este agente externo "guía" el entrenamiento de la red mediante una comparación entre las salidas deseadas (*targets*) y la salidas que proporciona la red, tal como muestra la siguiente ilustración:

[8] LI, Deng. DONG, Yu. Deep Learning: Methods and Applications [en línea]. Foundations and Trends® in Signal Processing. Now Publish, 2014 p. 200. [Consultado 20 de septiembre, 2014]. Disponible en Internet: http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf
[9] HERNANDEZ, Op. cit.
[10] SCHMIDHUBER, Op. cit.
[11] LÓPEZ S., Op. cit. p 13.

Figura 1. Aprendizaje supervisado



Fuente: LÓPEZ S, Jesús A. CAICEDO B, Eduardo F. "Una aproximación práctica a las redes neuronales artificiales". Editorial Universidad Del Valle, 2009. p 15.

En la figura anterior, el "mundo real" es referido a un conjunto de datos de entrada y salida, el cual corresponde al problema en específico a modelar.

- **No supervisado:** El aprendizaje es realizado presentándole a la red los datos directamente, es decir, ahora no existe un agente supervisando el entrenamiento. En este caso, la red aprende los datos de la entrada modificando los pesos en función de los datos caracterizados formando, en algunos casos, *clusters* o agrupación de los datos, tendiendo a clasificar los datos de forma probabilística.

Figura 2. Aprendizaje no supervisado

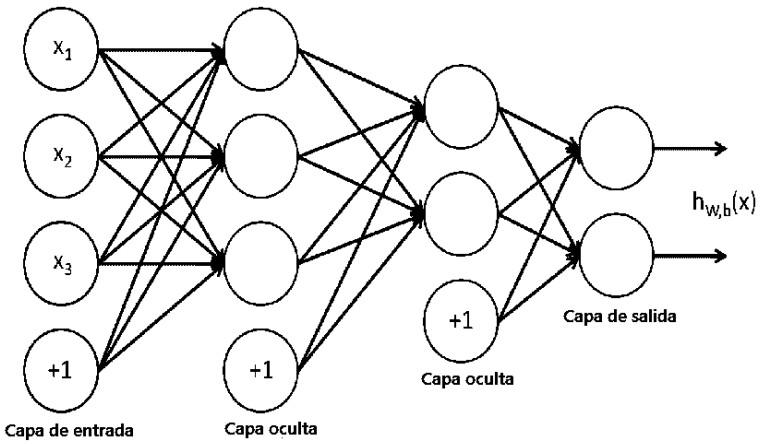
Fuente: LÓPEZ S, Jesús A. CAICEDO B, Eduardo F. "Una aproximación práctica a las redes neuronales artificiales". Editorial Universidad Del Valle, 2009. p 16.

- **Híbrido:** En las arquitecturas del *deep learning*, algunas redes poseen o utilizan ambos tipos de entrenamientos, ya sea comenzando con un pre-entrenamiento supervisado y finalizando con uno no supervisado o viceversa. Esto es con el fin de lograr un ajuste fino[12], disminuir el tiempo de convergencia, entre otras funcionalidades.

Las arquitecturas y algoritmos de aprendizaje del *deep learning* se basan, en su mayoría, en las utilizadas por RNA "comunes", como los son las siguientes:

4.2.1. Backpropagation. Es un algoritmo de entrenamiento el cual consiste en propagar el error de la capa de salida hacia las capas ocultas, lo cual da referencia a su traducción al español "retropropagación"[13]. Por lo que el error de la capa de salida (pues este algoritmo se aplica, por lo general, a RNA multicapa) se propaga hacia atrás, para estimar el error en las salidas de las capas ocultas, cuyo fin es calcular los pesos sinápticos de las neuronas en dichas capas ocultas. El método de aprendizaje es supervisado.

Figura 3. MLP (Perceptrón multicapa). Un ejemplo muy común de una RNA multicapa



Fuente: NG, Andrew. Sparse Autoencoder [en línea]. CS294A Lecture notes, 72 (2011). p 5. [Consultado 10 de septiembre, 2014]. Disponible en Internet: http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

[12] LI Deng; DONG Yu. Op. cit. p 241.
[13] LÓPEZ S, Op. cit. p 57.

En el algoritmo de entrenamiento se utiliza el concepto de gradiente descendente, de tal forma que se encuentre el punto en que el error de aprendizaje sea mínimo. Sin embargo, la búsqueda del mínimo en la superficie de error tiene variantes, al igual que el algoritmo de aprendizaje, como por ejemplo, gradiente descendente con alfa variable, con gradiente conjugado o el algoritmo de Levenberg Marquardt, que es una variante optimizada del algoritmo. De esta manera, se explica el algoritmo básico; los pasos este son:

- **Paso 1:** Comenzar con unos pesos aleatorios pequeños
- **Paso 2:** Mientras la condición de parada sea falsa se ejecutan los pasos 3 al 12
- **Paso 3:** Se aplica un vector de entrada

$$x_p = \left[x_{p1}, x_{p2}, \dots, x_{pn}\right]^T$$

- **Paso 4:** Se calcula los valores de las entradas netas para la capa oculta

$$Net_{pj}{}^h = \sum_{i=1}^{N} x_{pi}\, w_{ji}{}^h + \theta_j{}^h$$

- **Paso 5:** Se calcula la salida de la capa oculta

$$i_{pj}{}^h = f^h{}_j\left(Neta_{pj}{}^h\right)$$

- **Paso 6:** Se calcula los valores netos de entrada para la capa de salida

$$Net_{pk}{}^o = \sum_{j=1}^{L} x_{pj}{}^o\, w_{kj}{}^o + \theta_j{}^o$$

- **Paso 7:** Se calcula la salida de la red

$$y_{pk} = f_k{}^o(Neta_{pk}{}^o)$$

- **Paso 8:** Se calcula los términos de error para las unidades de salida

$$\delta_{pk}{}^o = \left(d_{pk} - y_{pk}{}^o\right) f_k{}^o(Neta_{pk}{}^o)$$

- **Paso 9:** Se estima los términos de error para las neuronas ocultas

$$\delta_{pj}{}^h = f_j{}^h\left(Net_{pj}{}^h\right) \sum_{k=1}^{M} \delta_{pk}{}^o\, w_{kj}{}^o$$

- **Paso 10:** Se actualiza los pesos en la capa de salida

$$w_{kj}{}^o(t+1) = w_{kj}{}^o(t) + \alpha \delta_{pk}{}^o i_{pj}{}^h$$

- **Paso 11:** Se actualiza los pesos de la capa oculta

$$w_{kj}{}^h(t+1) = w_{kj}{}^h(t) + \alpha \delta_{pk}{}^h x_{pi}$$

- **Paso 13:** Se verifica si el error global cumple con la condición de finalizar

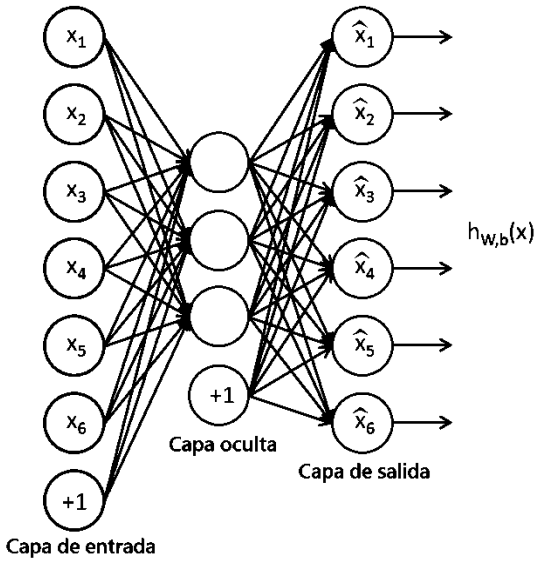
$$E_P = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{M}\left(d_{pk} - y_{pk}\right)^2$$

La nomenclatura para el algoritmo es:

x_p	Patrón o vector de entrada.
x_{pi}	Entrada i-ésima del vector de entrada x_p.
N	Dimensión del vector de entrada.
P	Número de ejemplos, vectores de entrada y salidas diferentes.
L	Número de neuronas de la capa oculta: h
M	Número de neuronas de la capa de salida, dimensión del vector de salida.
${w_{ji}}^h$	Peso de interconexión entre la neurona i-ésima de la entrada y la j-ésima de la capa oculta.
${\theta_j}^h$	Término de tendencia de la neurona j-ésima de la capa oculta.
${Neta_{pk}}^o$	Entrada neta de la k-ésima neurona de la capa de salida.
y_{pk}	Salida de la k-ésima unidad de salida.
${f_k}^o$	Función de activación de la k-ésima unidad de salida $0 \in \mathbb{R}$.
d_{pk}	Valor de salida deseado para la k-ésima neurona de la capa de salida.
E_P	Valor del error para el p-ésima patrón de aprendizaje.
α	Taza o velocidad de aprendizaje.
${\delta_{pk}}^o$	Término de error para la k-ésima neurona de la capa de salida.
${\delta_{pj}}^h$	Término de error para la j-ésima neurona de la capa oculta h.
${f_j}^h$	Derivada de la función de activación de la j-ésima neurona de la capa oculta.
${f_j}^h$	Derivada de la función de activación de la k-ésima neurona de la capa de salida.

4.2.2. Autoencoder. Es una RNA con una arquitectura muy parecida a la del MLP, sin embargo, tienen ciertas diferencias. Las diferencias más destacables se encuentran en la arquitectura, la cual varia de la MLP porque en su capa oculta presenta menos neuronas que en la capa de salida, como se muestra en la siguiente figura.

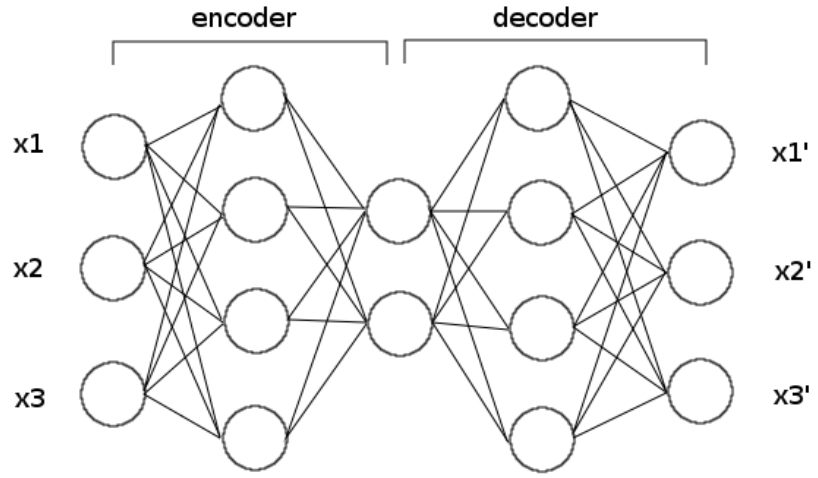
Figura 4. Arquitectura de un Autoencoder



Fuente: NG, Andrew. Sparse Autoencoder [en línea]. CS294A Lecture notes, 72 (2011). p 13. [Consultado 10 de septiembre, 2014]. Disponible en Internet: http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

Esta RNA puede tener varias capas ocultas, pero se trata de que la cantidad de capas ocultas sea impar, de tal forma que se guarde simetría, como se muestra en la siguiente figura.

Figura 5. Autoencoder de más de una capa oculta



Fuente: GRAFF, P. FEROZ, F. HOBSON, M. P. LASENBY, A. SKYNET: An efficient and robust neural network training tool for machine learning in astronomy [en línea]. Monthly Notices of the Royal Astronomical Society 441 (2014). p. 4. [Consultado 15 de septiembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1309.0790v2.pdf

Otra diferencia, que también se encuentra en la arquitectura, radica en que las neuronas de salida son de igual cantidad que las de entrada, como se muestra en las anteriores figuras.

El nombre de la red viene de su funcionamiento, ya que la red tiene como 'objetivo' reconstruir las entradas, es decir, la salida de la red es una reconstrucción de las entradas. La red, al tener menos neuronas en las capas ocultas, obliga a los datos de las entradas a comprimirse y a reducir la dimensión de los datos, de aquí el nombre de la red, de forma similar a lo que lo haría un análisis de componentes principales (PCA).

La última diferencia es el entrenamiento. A pesar de que la red utiliza el algoritmo de *backpropagation*, el algoritmo es no supervisado. Como se ha mencionado, la red recibe los datos y los comprime (*encoder*) y los regresa a sus dimensiones (decoder), obteniendo una representación con dimensiones más pequeñas hasta tener una aproximación a la entrada.

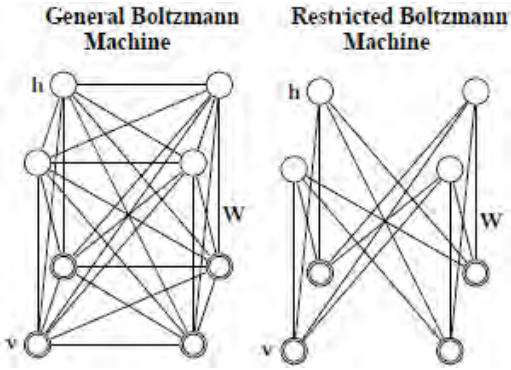
En el algoritmo de entrenamiento se puede utilizar cualquier derivación del *backpropagation*, descritas anteriormente. Se aplica el algoritmo actualizando los pesos teniendo en cuenta que las salidas deseadas son las mismas entradas, en base al error global.

4.2.3. Restricted Boltzmann Machine (RBM). Es una RNA estocástica, lo que hace que sus funciones de activación tengan un comportamiento no determinístico, ayudando a la red a salir de un mínimo local. Al ser la red estocástica, los datos admitidos por la red están en el rango de 0 y 1, por lo que se dice que la red posee unidades binarias.

Ahora bien, el nombre indica que la presente red es una variación de la red *Boltzmann Machine* (BM). La variación de la red se debe a que las conexiones entre las neuronas, tanto de la capa oculta como de entrada o visible, están conectadas, como se muestra en la siguiente figura. En la RBM, dichas conexiones ya no existen, debido a que el entrenamiento de una BM podía llegar a ser muy tedioso, tanto en tiempo como computacionalmente.

El tiempo de entrenamiento de la BM crece exponencialmente a medida que la red se hacía más grande, lo que también implica un gran número de conexiones y el equilibrio al que debía llegar la red se veía afectado. Al manejar las conexiones o restringirlas, como se muestra en la siguiente figura, se hace que la red pueda manejar de mejor forma una estructura más grande y pueda llegar a estabilizarse en un mínimo local de la gráfica de error global de la red.

Figura 6. Comparación de arquitecturas entre la Boltzmann Machine y Restricted Boltzmann Machine

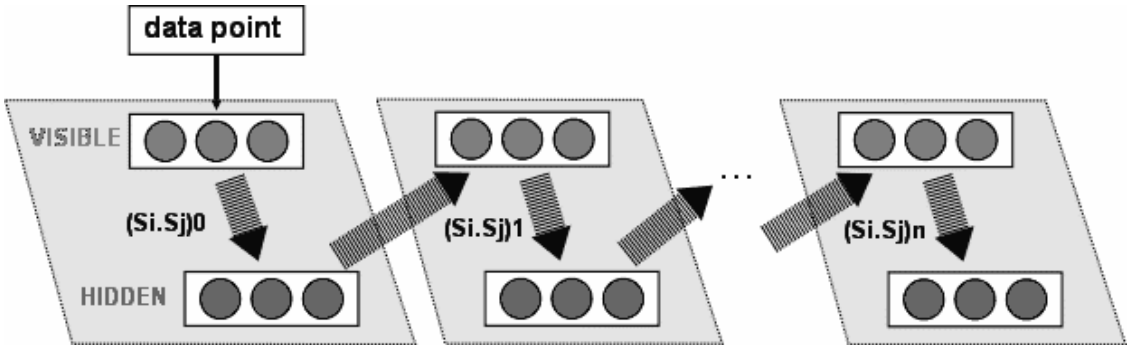


Fuente: SALAKHUTDINOV, Ruslan; HINTON, Geoffrey E. Deep boltzmann machines [en línea]. En International Conference on Artificial Intelligence and Statistics. 2009. p 2. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS09_SalakhutdinovH.pdf

En la figura anterior, también se puede apreciar la arquitectura que posee la RNA. La arquitectura la forman capas de unidades ocultas y visibles, que se denotan en dicha figura con las letras *h* y *v*, respectivamente. Esta red solo posee estas capas y se definen la cantidad de unidades que cada una poseen, obviamente, dependiendo de la aplicación.

A diferencia de las anteriores arquitecturas, esta red no es *feedforward*, es decir, la información en la red no se propaga sola dirección; en este caso, la red propaga la información en desde la capa visible hasta la capa oculta y viceversa. Sin embargo, se puede pensar que los pesos pueden cambiar dependiendo de la dirección, pero no es así, pues los pesos son los mismos, independiente de que capa este actualizando sus pesos, como muestra la siguiente figura.

Figura 7. Entrenamiento de una RBM



Fuente: Restricted Boltzmann Machine – Short Tutorial [en línea]. iMONAD. [Consultado 13 de enero, 2015]. Disponible en Internet: http://imonad.com/rbm/restricted-boltzmann-machine/

El tipo entrenamiento de la red depende de la aplicación, es decir, el entrenamiento de la red puede ser supervisado o no supervisado. Sin embargo, es más usado el entrenamiento no supervisado, por lo que se hará énfasis en él. Cabe rescatar que otra diferencia entre la RB y la RBM se encuentra en el algoritmo de entrenamiento. La RB utiliza una técnica o algoritmo denominado *simulated annealing* o temple simulado, el cual busca que la red tenga la mínima "energía" posible y se encuentre en un estado de equilibrio térmico, pero no se entrara en detalles con este concepto. A diferencia de las RBs, las RBMs no utiliza el anterior algoritmo[14].

Como se ha mencionado anteriormente, al poseer dos tipos entrenamiento, las RBMs poseen varios algoritmos de entrenamiento. Sin embargo, se explica el algoritmo más utilizado en el entrenamiento no supervisado.

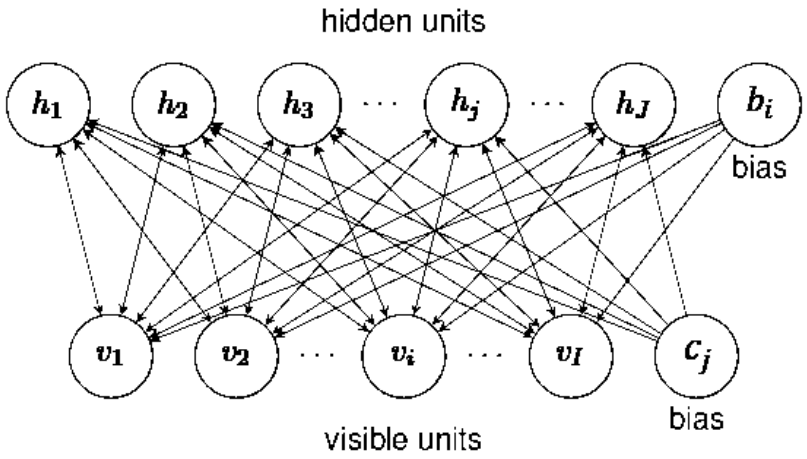
La arquitectura de la red también es llamada y considerada un modelo en grafo*,[15], más especialmente, un modelo conocido como *Markov random fields* (MRF). Si se asume que el modelo de grafo es conocido y la función de energía pertenece a un conjunto de funciones parametrizas θ, el aprendizaje no supervisado de una

[14] LI, Yue. Review of Boltzmann Machine and simulated annealing [en línea]. CSC321 Tutorial 9. p 7. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://www.cs.utoronto.ca/~yueli/CSC321_UTM_2014_files/tut9.pdf

[15] KOLLER, Daphne; FRIEDMAN, Nir. Probabilistic graphical models: principles and techniques [en línea]. MIT press, 2009. p. 1. [Consultado 15 de octubre, 2014]. Disponible en Internet: http://mitpress.mit.edu/sites/default/files/titles/content/9780262013192_sch_0001.pdf

distribución de datos MRF puede ajustar los parámetros θ. Se define como $p(x|\theta)$ para hacer hincapié en la dependencia de una distribución de sus parámetros[16].

Figura 8. Arquitectura de una RBM con j y i unidades ocultas y visibles, respectivamente



Fuente: LOPES, Noel; RIBEIRO, Bernardete; GONÇALVES, Joao. Restricted Boltzmann machines and deep belief networks on multi-core processors [en línea]. En Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE, 2012. p. 8. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6252431&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6252431

Se asume que los datos de entrenamiento son independientes e idénticamente distribuidos. Una forma de estimar los parámetros de un modelo estocástico o estadístico es estimando la máxima verosimilitud*. Para ello, se hace uso de los MRF, ya que al encontrar los parámetros que maximizan los datos de entrenamiento bajo una distribución MRF, equivale a encontrar los parámetros θ que maximizan la verosimilitud de los datos de entrenamiento[17]. Maximizar dicha verosimilitud es el objetivo que persigue el algoritmo de entrenamiento de una RBM.

---

[16] FISCHER, Asja; IGEL, Christian. An introduction to restricted Boltzmann machines [en línea]. En Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Berlin Heidelberg, 2012. p. 2. [Consultado 21 de septiembre, 2014]. Disponible en Internet: http://image.diku.dk/igel/paper/AItRBM-proof.pdf

* Es un modelo probabilístico grafico el cual describe la distribución de probabilidad mediante la asignación de propiedades de dependencia e independencia condicional entre variables aleatorias en una estructura gráfica.

[17] FISCHER, Op. cit. p. 18.

A pesar de utilizar la distribución MRF y sus consideraciones, se llega a ecuaciones inviables de implementar, computacionalmente hablando[18,19]. Un simplificado desarrollo de las expresiones obtenidas se muestra a continuación.

$$E(v,h|\theta) = -\sum_{ij} h_j w_{ij} v_i - \sum_i b_i v_i - \sum_j c_j h_j$$

$$p(v|\theta) = \prod_{n=1}^{N} \sum_{\boldsymbol{h}} p(v,h|\theta) = \prod_{n=1}^{N} \frac{\sum_{\boldsymbol{h}} e^{-E(v,h|\theta)}}{\sum_{\boldsymbol{h},\boldsymbol{v}} e^{-E(v,h|\theta)}}$$

$$\log p(v|\theta) = \sum_{n=1}^{N} \left(\log \sum_{\boldsymbol{h}} e^{-E(v,h|\theta)} - \log \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(v,h|\theta)} \right)$$

$$\frac{\partial \log p(v|\theta)}{\partial w_{ij}} = \sum_{n=1}^{N} \left(v_i \sum_{\boldsymbol{h}} h_j p(\boldsymbol{h}|\boldsymbol{v}) - \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j p(\boldsymbol{h},\boldsymbol{v}) \right)$$

$$\frac{\partial \log p(v|\theta)}{\partial w_{ij}} = < v_i h_j >_{datos(0)} - < v_i h_j >_{modelo(\infty)}$$

La expresión <.> denota la esperanza para de la distribución, ya sea de los datos o de los modelos.

Para evitar el problema anterior, las esperanzas que se obtienen de MRF pueden ser aproximadas por muestras extraídas de distribuciones basadas en las técnicas de Markov Chain Monte Carlo Techniques (MCMC). Las técnicas de MCMC utilizan un algoritmo denominado *Gibbs sampling,* con el cual se obtiene una secuencia de observaciones o muestras que se aproximan a partir de una distribución de verosimilitud de múltiples variables aleatorias. La idea básica de este algoritmo es actualizar cada variable posteriormente en base a su distribución condicional dado el estado de las otras variables.

El algoritmo *Gibbs sampling* busca estimar las esperanzas tomando muestras; en el caso de un nuevo estado de la capa ***h*** se basa en la distribución $p(h|v)$ y para ***v***, con un estado de la distribución de $p(v|h)$. Sin embargo, se presenta un

---

[18] Ibíd., p. 20.

[19] HINTON, Geoffrey; OSINDERO, Simon; TEH, Yee-Whye. A fast learning algorithm for deep belief nets [en línea]. Neural computation, 2006, vol. 18, no 7, p. 3. [Consultado 01 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~hinton/absps/fastnc.pdf

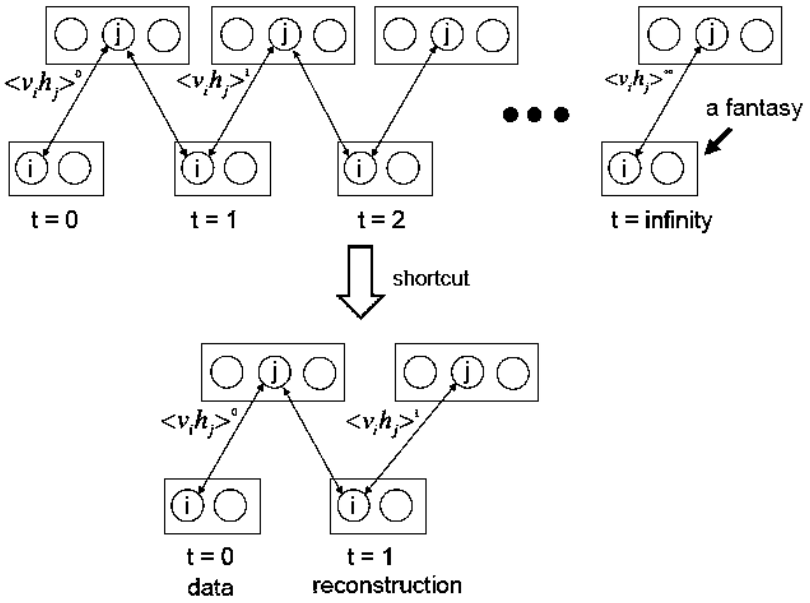
* La verosimilitud o la función de verosimilitud es una función en base de los parámetros de un modelo estadístico, el cual permite realizar inferencias acerca de su valor.

problema, pues se necesita que las iteraciones del algoritmo sean infinitos, pero se llega a una aproximación, la cual da resultados aceptables[20,21].

$$\frac{\partial \log p(v^0)}{\partial w_{ij}} = < h_j^0(v_i^0 - v_i^1) > + < v_i^1(h_j^0 - h_j^1) > + < h_j^1(v_i^1 - v_i^2) > + \cdots$$

$$\frac{\partial \log p(v^0)}{\partial w_{ij}} \approx < v_i^0 h_j^0 > - < v_i^\infty h_j^\infty > \approx < v_i^0 h_j^0 > - < v_i^1 h_j^1 >$$

Figura 9. Aproximación por el algoritmo de Gibbs sampling



Fuente: LI, Yue. Review of Boltzmann Machine and simulated annealing [en línea]. CSC321 Tutorial 9. p. 11. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://www.cs.utoronto.ca/~yueli/CSC321_UTM_2014_files/tut9.pdf

A pesar de las consideraciones realizadas para la implementación computacional, el algoritmo puede no llegar a la máxima verosimilitud, por lo que es necesario, al igual que en el algoritmo de entrenamiento de *backpropagation,* utilizar el gradiente descendente. Uno de los algoritmos que permite esto es denominado *contrastive divergence* (CD-k), por el que son entrenadas las RBMs usualmente, como se explica a continuación.

[20] Ibíd., p. 4.
[21] LI Yue, Op. cit. p. 11.

- **Paso 1:** Inicializar $\Delta w_{ij} = \Delta b_i = \Delta c_i = 0$
- **Paso 2:** Para todo $v \in S$ hacer
- **Paso 3:** $v^{(0)} \leftarrow v$
- **Paso 4:** Para $t = 0, \dots, k-1$ hacer
- **Paso 5:** Para $j = 0, \dots, n$ hacer muestras $h_j^{(t)} \sim p\left(h_j | v^{(t)}\right)$
- **Paso 6:** Para $i = 0, \dots, m$ hacer muestras $v_i^{(t+1)} \sim p\left(v_i | h^{(t)}\right)$
- Fin del ciclo del paso 4
- **Paso 7:** Para $i = 1, \dots, m, j = 1, \dots, n$ hacer
- **Paso 8:** $\Delta w_{ij} \leftarrow \Delta w_{ij} + p\left(h_j = 1 | v^{(0)}\right) v_j^{(0)} - p\left(h_j = 1 | v^{(k)}\right) v_j^{(k)}$
- **Paso 9:** $\Delta b_i \leftarrow \Delta b_i + v_j^{(0)} - v_j^{(k)}$
- **Paso 10:** $\Delta c_i \leftarrow \Delta c_i + p\left(h_j = 1 | v^{(0)}\right) - p\left(h_j = 1 | v^{(k)}\right)$
- Fin del ciclo del paso 7
- Fin del ciclo del paso 2

En el algoritmo se le denomina a S como los datos de entrenamiento, k como la cantidad de datos de entrenamiento y a t como el valor de la iteración. Cabe rescatar que la salida o visualización de la red se hace mediante los pesos, tanto de la capa oculta como visible, y los *bias* (b_i y c_i).

En el algoritmo aparecen ciertas ecuaciones no expresadas anteriormente. De esta forma las ecuaciones desarrolladas y necesarias son:

$$p\left(h_j = 1 | v\right) = \sigma\left(\sum_i w_{ij} v_i + c_j\right)$$

Donde,

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

La anterior ecuación se utilizan para en el muestre con unidades binarias. Para las unidades que no lo son, es decir, son reales se utiliza la siguiente expresión:

$$p\left(h_j | v\right) = \frac{e^{h_j (c_j + w_j v)}}{\sum_h e^{h_j (c_j + w_j v)}}$$

De forma análoga para las unidades visibles condicionadas con la capa oculta.

$$p(v_i = 1|h) = \sigma\left(\sum_j w_{ij}h_j + b_i\right)$$

$$p(v_i|h) = \frac{e^{v_i(b_i+w_ih)}}{\sum_v e^{v_i(b_i+w_ih)}}$$

4.3. ARQUITECTURAS MÁS REPRESENTATIVAS

Estas arquitecturas fueron entre los primeros conceptos que se generaron de *deep learning,* y de las cuales se basan para generar otras redes.

4.3.1 Deep Belief Network (DBN). Una DBN se considera como un "apilamiento" de redes RBM[22], es decir, las capas que conforman una DBN son RBMs[23]. Por lo anterior, esta red también es un modelo en grafo estocástico, pero esta vez, constituido por una jerarquía de variables latentes*,[24]. Cada capa de RBMs extrae un nivel de abstracción de características de los datos de entrenamiento, cada vez más significativa; pero para ello, la capa siguiente necesita la información de la capa anterior para "abstraer" las características, lo que implica el uso de las variables latentes y también por lo que se denomina una arquitectura jerárquica, lo cual es distintivo del *deep learning*.

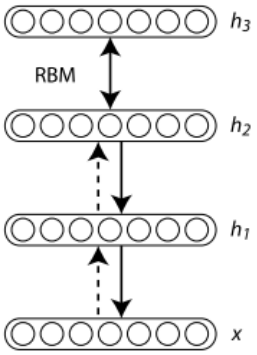
[22] BENBIO, Yoshua. Learning deep architectures for AI [en línea]. Technical Report 1312, Université de Montréal, 2007. p. 25. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf

[23] AREL, Itamar; ROSE, Derek C.; KARNOWSKI, Thomas P. Deep machine learning-a new frontier in artificial intelligence research [en línea]. Computational Intelligence Magazine, IEEE, 2010, vol. 5, no 4, p. 17. [Consultado 07 de octubre, 2014]. Disponible en Internet: http://msrds.googlecode.com/svn/trunk/rbm/DML_Arel_2010.pdf

* Estas variables no son visualizadas directamente, pero son variables que son inferidas a través de las variables que si son observables a partir de modelos matemáticos.

[24] BENGIO, Yoshua. Learning deep architectures for AI [en línea]. Foundations and trends® in Machine Learning, 2009, vol. 2, no 1, p. 21. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf

Figura 10. Arquitectura de una DBN



Fuente: BENGIO, Yoshua. Learning deep architectures for AI [en línea]. Foundations and trends® in Machine Learning, 2009, vol. 2, no 1, p. 37. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf

Como se muestra en la anterior figura, la arquitectura de una DBN consiste en varias RBMs apiladas. Ahora bien, el entrenamiento de una DBN infiere varias formas de realizarlo, al igual que en una RBM, depende de la aplicación a realizar. El entrenamiento puede ser de tipo no supervisado es utilizado para la reconstrucción de las entradas o datos de entrenamiento, mientras que si se utiliza un entrenamiento supervisado, la aplicación a realizar es de clasificación[25].

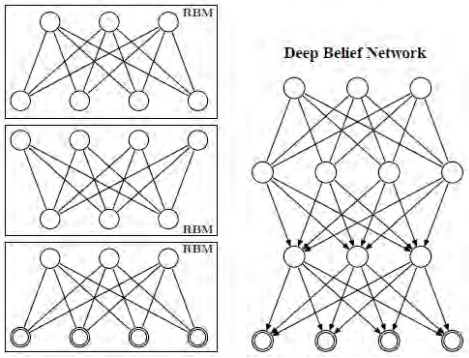
El entrenamiento de una red DBN también puede ser hibrido, es decir, para este caso, primero comenzar con un entrenamiento no supervisado y después realizar una *backpropagation* o entrenamiento supervisado para un "ajuste fino". No obstante, una DBN tiene varias formas de ser entrenada[26,27].

Un algoritmo comúnmente utilizado es denominado "g*reedy layer-wise*", el cual se desarrolla de una forma "simple", debido a que se entrena la DBN a partir de entrenar RBMs. La idea básica del algoritmo es que se apilan las capas, de tal forma que se obtenga una RBM, por lo que primero se entrena la RBM formada en

[25] HINTON Geoffrey. A fast learning algorithm for deep belief nets, Op. cit. p. 1.
[26] BENGIO, Yoshua, et al. Greedy layer-wise training of deep networks [en línea]. Advances in neural information processing systems, 2007, vol. 19, p. 153. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2006_739.pdf
[27] SALAKHUTDINOV, Ruslan. Learning Deep Generative Models [en línea]. Ph. D. Thesis. Department of Electrical and Computer Engineering, University of Toronto. 2009. p. 1-84. [Consultado 24 de septiembre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/papers/Russ_thesis.pdf

la capa inferior, que contiene los datos, la cual brindara los pesos y las entradas a la siguiente RBM formada o, en otros términos, se entrena una RBM y la capa oculta de la capa RBM ya entrenada, pasa a ser la capa visible de la siguiente, como se muestra en la siguiente ilustración.

Figura 11. Entrenamiento de una DBN. Izquierda: Algoritmo aplicado (greedy layer-wise) para entrenar una DBN. Derecha: DBN correspondiente



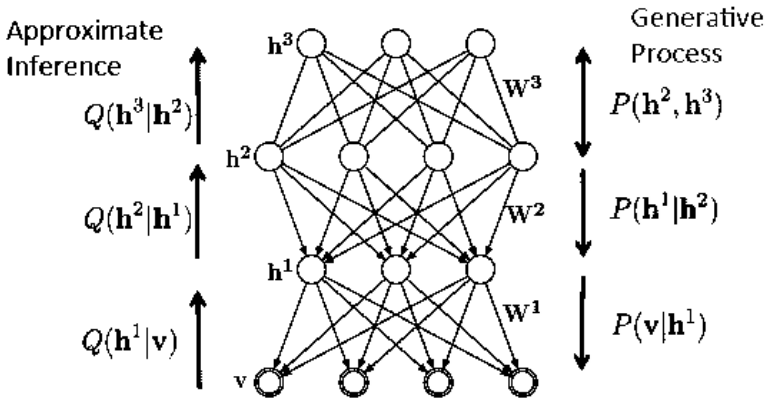
Fuente: SALAKHUTDINOV, Ruslan. Learning Deep Generative Models [en línea]. Ph. D. Thesis. Department of Electrical and Computer Engineering, University of Toronto. 2009. p. 9. [Consultado 24 de septiembre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/papers/Russ_thesis.pdf

Cabe destacar que el algoritmo no es supervisado. De esta manera, el algoritmo a emplear es el siguiente:

- **Paso 1:** Inicializar $b^0 = 0$
- **Paso 2:** Para $l = 1$ hasta L hacer
- **Paso 3:** Inicializar $w^i = 0, b^i = 0$
- **Paso 4:** Mientras *criterio* hacer
- **Paso 5:** Muestrear $g^0 = x$ desde $\hat{p}$
- **Paso 6:** Para $i = 1$ hasta $l - 1$ hacer
- **Paso 7:** Muestrear g^i desde $Q(g^i|g^{i-1})$
- Fin del ciclo del paso 6
- **Paso 8:** Entrenar la RBM formada
- Fin del ciclo del paso 4
- Fin del ciclo del paso 2

En el algoritmo se denota a $\hat{p}$ como la distribución de la entrada o datos de entrenamiento de la red, L es el número de capas a entrenar, w^i es la matriz de pesos del nivel i, para i desde 1 hasta L, b^i es el vector de *bias* para el nivel i hasta L; $Q(a|b)$ es una aproximación de la probabilidad tomada de la probabilidad condicionada, debido a que el algoritmo necesita ejercitarse por partes y no se tiene una probabilidad previa, como se muestra a continuación:

Figura 12. Aproximación realizada en el algoritmo de entrenamiento DBN



Fuente: SALAKHUTDINOV, Ruslan. Deep Learning, KDD Tutorial [en línea]. Department of Computer Science. Department of Statistics. University of Toronto. Canadian Institute for Advanced Research (CIFAR). 2014. p. 78. [Consultado 20 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/talk_KDD_part1_pdf.pdf

En donde $Q(h^t|h^{t-1}) = \prod_j \sigma(\sum_i w^t\, h_i^{t-1})$, mientras que $P(h^{t-1}|h^t) = \prod_j \sigma(\sum_i w^t\, h_i^t)$. Finalmente, $\hat{p}$ se define:

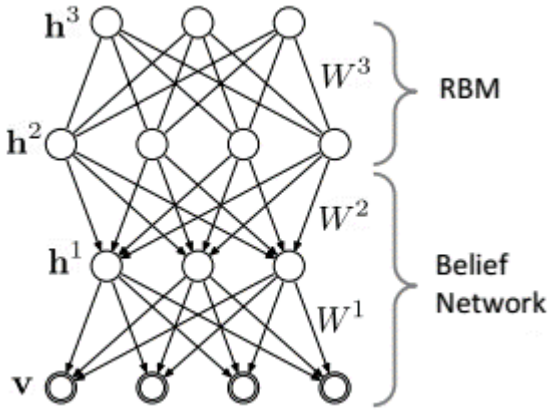
$$\hat{p}(g^l) = \sum_{g^{l-1}} \hat{p}^{l-1}(g^{l-1}) Q\left(g^i|g^{i-1}\right)$$

En la anterior ecuación, se le denomina a g^i como las variables ocultas en la capa i; estas son w, h y v.

A medida que la red se va entrenando, para el caso propuesto de tres capas ocultas, las últimas dos capas en la parte superior forman un grafo no dirigido (RBM) y las capas restantes forman una red *belief network** con conexiones de

arriba hacia abajo[28], como se muestra en la siguiente figura, de allí se deriva el nombre de la red.

Figura 13. Formación de la red Belief Network en el entrenamiento de una DBN



Fuente: SALAKHUTDINOV, Ruslan. Deep Learning, KDD Tutorial [en línea]. Department of Computer Science. Department of Statistics. University of Toronto. Canadian Institute for Advanced Research (CIFAR). 2014. p. 77. [Consultado 20 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/talk_KDD_part1_pdf.pdf

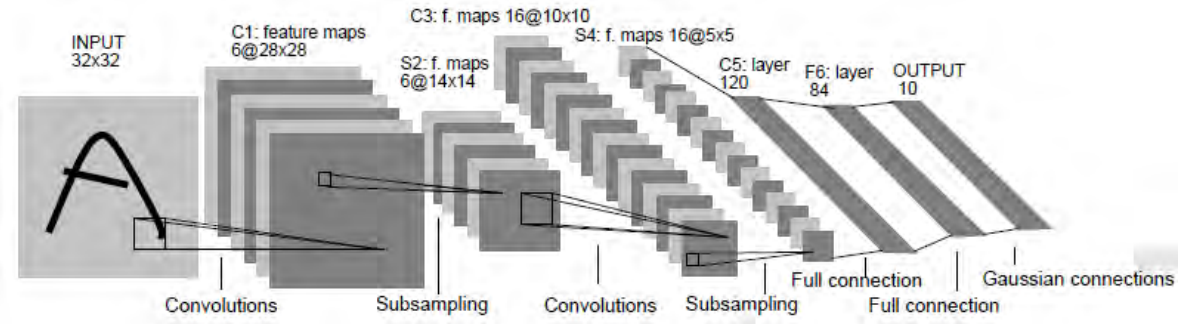
Al entrenar a las DBNs, se puede notar que también se necesita entrenar a las RBMs varias veces, por lo el algoritmo de entrenamiento de las RBMs también se puede considerar como otro parámetro dentro del entrenamiento de una DBN. Cabe aclarar que el algoritmo anterior hace uso de *contrastive divergence* como algoritmo de entrenamiento para las RBMs.

[28] HINTON, Geoffrey E.; SALAKHUTDINOV, Ruslan R. A better way to pretrain deep Boltzmann machines [en línea]. En Advances in Neural Information Processing Systems. 2012. p. 3. [Consultado 11 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~fritz/absps/DBM_pretrain.pdf

* Una red belief net o red de Bayes, son grafos dirigidos acíclicos que representan a variables aleatorias, entre las cuales se presenta una relación.

4.3.2. Convolutional Neural Network (CNN). Una CNN es una red multicapa jerárquica, por lo cual, al igual que la DBN, extrae características en sus capas ocultas. El nombre de la red se deriva de su funcionamiento y su arquitectura, ya que las CNNs se conforman desde 5 hasta 7 capas[29]. Dichas capas se organizan en capas convolucionales y de sub-muestreo (*subsampling*)[30], como se muestra en la siguiente figura.

Figura 14. Arquitectura de una CNN



Fuente: LECUN, Yann, et al. Gradient-based learning applied to document recognition [en línea]. Proceedings of the IEEE, 1998, vol. 86, no 11, p. 7. [Consultado 2 de octubre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Las CNNs fueron inspiradas por la estructura del sistema de visión[31], en las cuales se encuentran capas alternadas de células simples, que se encargan de la extracción de características, y de células complejas, que se encargan de la abstracción o identificación del objeto; en términos de una CNN, esta última capa es la capa de sub-muestreo[32].

[29] BENGIO, Yoshua. Learning deep architectures for AI, Op. cit. p. 43.
[30] LECUN, Yann, et al. Gradient-based learning applied to document recognition [en línea]. Proceedings of the IEEE, 1998, vol. 86, no 11, p. 7. [Consultado 2 de octubre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
[31] HUBEL, David H.; WIESEL, Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [en línea]. The Journal of physiology, 1962, vol. 160, no 1, p. 106-154. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1962.sp006837/pdf
[32] WAGNER, Raimar, et al. Learning convolutional neural networks from few samples [en línea]. En Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, 2013. p. 1-7. [Consultado 11 de octubre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6706969&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6691896%2F6706705%2F06706969.pdf%3Farnumber%3D6706969

Consecuente a lo anterior, se han obtenido buenos resultados en aplicaciones del campo de visión por computador y el procesamiento de imágenes[33]. Por mencionar algunas, reconocimiento de dígitos escritos a mano[34,35] y clasificaciones de imágenes a partir de un gran conjunto de datos (*datasets*), como NORB[36] y CIFAR10[37]. El buen desempeño de esta red en imágenes se debe al grado de invariancia que la red posee[38], la cual también no posee una restricción en la dimensión en los datos de entrada o de entrenamiento; estos pueden tener una dimensión arbitraria.

El éxito de la CNN ha hecho que recientemente su estructura sea adecuada en otras redes[39], como en las RBMs[40] y DBNs[41]. Sin embargo, las capas o estructura de una red CNN, como se explicó anteriormente, consta de, junto con las dos etapas mencionadas, tres tipos de capas[42]: La capa convolucional, sub-muestreo

---

[33] CONG, Jason; XIAO, Bingjun. Minimizing Computation in Convolutional Neural Networks [en línea]. En Artificial Neural Networks and Machine Learning–ICANN 2014. Springer International Publishing, 2014. p. 281-290. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://vast.cs.ucla.edu/sites/default/files/publications/CNN_ICANN14.pdf

[34] LECUN Yann. Gradient-based learning applied to document recognition. 1998. Op. cit. p. 1.

[35] SIMARD, Patrice Y.; STEINKRAUS, Dave; PLATT, John C. Best practices for convolutional neural networks applied to visual document analysis [en línea]. En 2013 12th International Conference on Document Analysis and Recognition. IEEE Computer Society, 2003. p. 958-958. [Consultado 28 de septiembre, 2014]. Disponible en Internet: http://vnlab.ce.sharif.ir/courses/85-86/2/ce667/resources/root/15%20-%20Convolutional%20N.%20N./ICDAR03.pdf

[36] LECUN, Yann; HUANG, Fu Jie; BOTTOU, Leon. Learning methods for generic object recognition with invariance to pose and lighting [en línea]. En Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, 2004. p. II-97-104. [Consultado 28 de septiembre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1315150&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1315150

[37] KRIZHEVSKY, A. Learning multiple layers of features from tiny images [en línea]. Master's thesis, Computer Science Department, University of Toronto, 2009. p. 1. [Consultado 11 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

[38] BOUVRIE, Jake. Notes on convolutional neural networks [en línea]. Center for Biological and Computer Learning, Department of Brain and Cognitive Sciences. MIT, Cambridge, 2006. [Consultado 03 de octubre, 2014]. Disponible en Internet: http://cogprints.org/5869/1/cnn_tutorial.pdf

[39] BENGIO Yosgua, 2009. Op. cit. p. 44.

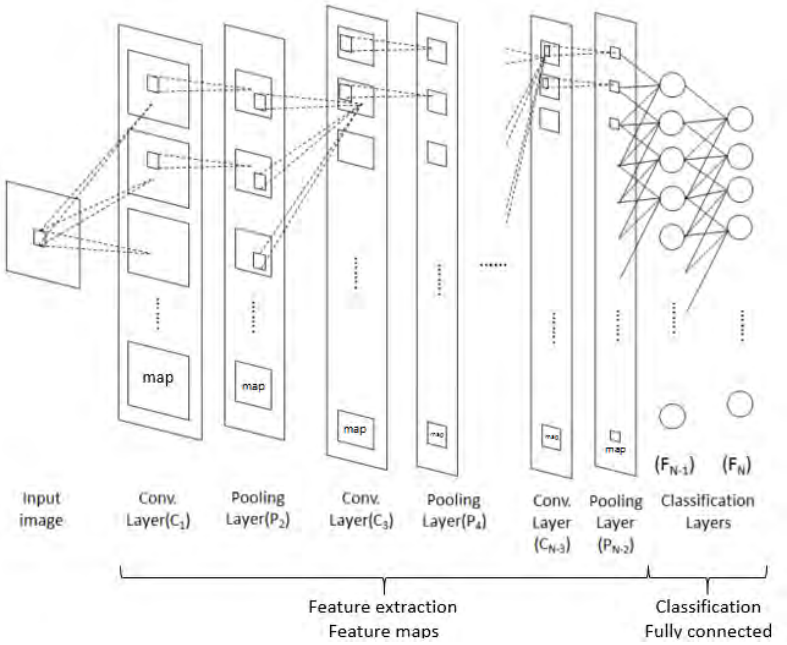
[40] DESJARDINS, Guillaume. Training deep convolutional architectures for vision [en línea]. Master Tesis. Departamento de informática e Investigación Operativa, Facultad de Artes y Ciencias, 2009. 116 p. [Consultado 20 de septiembre, 2014]. Disponible en Internet:https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/3646/Desjardins_Guillaume_2009_memoire.pdf?sequence=2

[41] LEE, Honglak, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations [en línea]. En Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009. p. 609-616. [Consultado 05 de octubre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.802&rep=rep1&type=pdf

[42] WAGNAR, Op cit. p. 2.

(*subsampling*) o *pooling* y la capa “totalmente conectada” o *fully connected*. Estas se muestran en la siguiente figura.

Figura 15. Arquitectura general, las etapas o capas que conforman una CNN



Fuente: XIE, Xiaohui; KIM, In-Jung. Handwritten Hangul recognition using deep convolutional neural networks [en línea]. International Journal on Document Analysis and Recognition (IJDAR), 2015, Vol. 18. Issue 1, p. 8. [Consultado 25 de septiembre, 2014]. Disponible en Internet: http://www.ics.uci.edu/~xhx/publications/HHR.pdf

Las CNNs se conforman de dos etapas: extracción de características y clasificación. En la etapa de extracción de características, como su nombre lo indica, se extraen las características que poseen los datos de entrada o entrenamiento (usualmente imágenes), formando los mapas de características (*feature maps*), los cuales van aumentando en número y disminuyendo en tamaño a medida que haya más capas ocultas en esta etapa. Además, esta consta de dos capas, la capa convolucional y la capa de *subsampling* o *pooling*, las dos capas van en cascada y, por lo general, no se suelen separar. A veces se suelen tratar a las dos capas (convolucionales y *subsampling*), como una sola[43].

[43] SCHERER, Dominik; SCHULZ, Hannes; BEHNKE, Sven. Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors [en línea]. En Artificial Neural Networks–ICANN 2010. Springer Berlin Heidelberg, 2010. p. 82-91. [Consultado 29 de septiembre, 2014]. Disponible en Internet: http://www.is.uni-bonn.de/papers/icann10_conv.pdf

Al ser las características propagadas, estas siguen siendo abstraídas y combinadas para producir unas características de "más alto nivel", a medida que el mapa de características o *feature maps* se van volviendo más pequeños[44]. El mencionado proceso continúa hasta tener mapas de características de muy baja resolución[45], por lo general, de una resolución de 1x1[46,47,48].

Una vez los mapas de características han sido extraídos, una red *fully connected* como clasificadora, ya sea un MLP, Radial Basis Function (RBF)[49], o arquitecturas un poco más desarrolladas y más eficientes[50]; se encarga de separar y clasificar cada característica. A pesar de que las CNNs varían en como las capas convolucionales y de muestreo o *pooling* son realizadas, también varian en cómo estas son entrenadas[51] , se explica el concepto general de cada una.

Capa convolucional: En analogía con las células simples de la corteza visual, esta capa extrae las características de los datos de entrenamiento por medio de la convolución entre dichos datos y filtros "entrenables" tipo *kernels*. Las características que se suelen encontrar en esta capa son bordes, esquinas y/o cruces[52].

---

[44] XIE, Xiaohui; KIM, In-Jung. Handwritten Hangul recognition using deep convolutional neural networks [en línea]. International Journal on Document Analysis and Recognition (IJDAR), 2015, Vol. 18. Issue 1, p. 1-13. [Consultado 25 de septiembre, 2014]. Disponible en Internet: http://www.ics.uci.edu/~xhx/publications/HHR.pdf

[45] SCHERER Dominik, Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors. Op. cit. p. 3.

[46] LECUN Yann, Gradient-based learning applied to document recognition, Op. cit. p. 8.

[47] CIRESAN, Dan C., et al. Flexible, high performance convolutional neural networks for image classification [en línea]. En IJCAI Proceedings-International Joint Conference on Artificial Intelligence. 2011. p. 1237. [Consultado 12 de octubre, 2014]. Disponible en Internet: http://people.idsia.ch/~juergen/ijcai2011.pdf

[48] MRAZOVA, Iveta; KUKACKA, Marek. Hybrid convolutional neural networks [en línea]. En Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. IEEE, 2008. p. 469-474. [Consultado 30 de septiembre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4618146&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4618146

[49] LECUN Yann, Gradient-based learning applied to document recognition, Op. cit. p. 9.

[50] TIVIVE, Fok Hing Chi; BOUZERDOUM, Abdesselam. Efficient training algorithms for a class of shunting inhibitory convolutional neural networks [en línea]. Neural Networks, IEEE Transactions on, 2005, vol. 16, no 3, p. 541-556. [Consultado 21 de septiembre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1427760&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1427760

[51] CIRESAN, Op. cit. p. 2.

[52] CONG, Op. cit. p. 2.

Como su nombre lo indica, la capa realiza la operación de convolución entre, ya sea la entrada (una imagen) o las características extraídas, y un *kernel*, el cual se compone de neuronas y pueden ser entrenables[53,54]. La ecuación que describe la salida de la red es la siguiente:

$$y_{i,j} = f\left(\sum_i k_j * x_i + b_j\right)$$

En donde el operador '$*$' denota la operación de convolución bidimensional. k_j son los pesos de los que se compone el *kernel* y b_j es el grado de libertad o *bias* que acompaña a cada *kernel* o filtro. x_i es la entrada o parte de la imagen, en el caso de la primera capa, o características, en caso de capas superiores. La función $f(x)$ es la función de activación, para la cual se recomienda en [40] una función tangente hiperbólica $f(x) = tanh\left(\frac{2}{3}x\right)$.

El tamaño del mapa de salida es definido por la siguiente ecuación, en donde M es la cantidad de mapas de una determinada capa de igual tamaño (M_x, M_y), (K_x, K_y) es el tamaño del *kernel* utilizado, (S_x, S_y) son los factores de desplazamiento que indican cuantos pixeles en x y en y se salta u omite entre cada convolución, teniendo en cuenta que el *kernel* siempre debe quedar dentro del mapa de características o imagen. n Es el índice de la capa presente.

$$M_x^n = \frac{M_x^{n-1} - K_x^n}{S_x^n + 1} + 1$$
$$M_y^n = \frac{M_y^{n-1} - K_y^n}{S_y^n} + 1$$

Las anteriores ecuaciones sirven para determinar el tamaño, largo como ancho, tanto de la capa presente, como de la capa *pooling*.

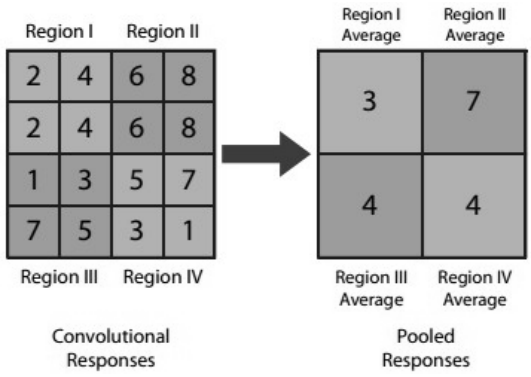
[53] WAGNER, Op. cit. p. 2.
[54] BOUVRIE, Op. cit. p. 3.

Capa pooling o subsampling: Esta capa es equivalente a las células complejas de la corteza visual y es usada para obtener una representación que es invariante contra pequeñas translaciones y distorsiones[55,56]. Dicha respuesta se logra mediante la combinación (*pooling*) de las respuestas obtenidas de la capa de convolución, consiguiendo un valor característico de las anteriores respuestas.

La salida de esta capa depende de la elección del usuario, debido a que se puede decidir qué operación realizar. Estas operaciones son: Promediar[57,58,59], *max-pooling*[60,61] o elegir el máximo valor, y *subsampling*[62,63,64,65] (consiste en utilizar una función de tangente hiperbólica). Estas operaciones se realizan en una región cuadrada, como se muestra en la siguiente figura, la cual no se puede solapar. A pesar de lo anterior, se ha encontrado que elegir el *max-pooling* en una región puede llevar a una más rápida convergencia, seleccionando las características invariantes superiores[66].

[55] WAGNER, Op. cit. p. 2.
[56] WU, David J. End-to-End Text Recognition with Convolutional Neural Networks [en línea]. Bachelor's Thesis with Honors and Distinction. Department of Computer Science. Stanford University, 2012. p. 13. [Consultado 07 de octubre, 2014]. Disponible en Internet: http://crypto.stanford.edu/~dwu4/papers/HonorThesis.pdf
[57] SCHERER Dominik. Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors, Op cit. p. 3.
[58] MRAZOVA, Op. cit. p. 2.
[59] WU, Op. cit. p. 12.
[60] XIE, Op. cit. 10.
[61] CIRESAN, 2011. Op. cit. p. 2.
[62] LECUN Yann. Gradient-based learning applied to document recognition, Op. cit. p. 6.
[63] CONG, Op. cit. p. 3.
[64] BOUVRIE, Op. cit. p. 4.
[65] SCHERER, Dominik; MÜLLER, Andreas; BEHNKE, Sven. Evaluation of pooling operations in convolutional architectures for object recognition [en línea]. En Artificial Neural Networks–ICANN 2010. Springer Berlin Heidelberg, 2010. p. 92-101. [Consultado 02 de octubre, 2014]. Disponible en Internet: http://www.ais.uni-bonn.de/papers/icann2010_maxpool.pdf
[66] Ibíd.

Figura 16. Ejemplo del proceso de pooling, en este caso, un promedio de una región de 2x2



Fuente: WU, David J. (2012) End-to-End Text Recognition with Convolutional Neural Networks [en línea]. Bachelor's Thesis with Honors and Distinction. Department of Computer Science. Stanford University. p. 13. Consultado 07 de octubre, 2014]. Disponible en Internet: http://crypto.stanford.edu/~dwu4/papers/HonorThesis.pdf

En la siguiente tabla se resume los tipos de *pooling* que se pueden aplicar en esta capa, con su respectiva implementación-salida.

Tabla 1. Resumen de las funciones aplicables en la capa de subsampling o pooling en una CNN

Tipo de *pooling*	**Implementación**	**Descripción**
Max-pooling	$\max_{rxr}(x_i)$	Se saca el máximo de una región cuadrada de tamaño rxr.
Promedio (*average*) o *downsampling*	$\text{mean}_{rxr}(x_i)$	Se saca el promedio de una región cuadrada de tamaño rxr.
Subsampling	$tanh\left(\beta \sum_{r,r} \frac{x_i}{r^2} + b\right)$	Se toma el promedio de la entradas de una región cuadrada de tamaño rxr y se multiplican por un escalar entrenable (β) y se le suma un entrenable *bias* (b) y se calcula el resultado a través de la función no lineal.

Capa fully connected: Esta es la última capa y etapa de una CNN y es la encargada de clasificar o etiquetar a cada clase de característica. Para ello, las anteriores capas han hecho que las características se encuentren en baja resolución, como se mencionó anteriormente, para ser estas quienes alimentan a una RNA totalmente conectada[67], de allí su nombre *fully connected*.

Se dice que las anteriores capas a estas no están totalmente conectadas debido a que la respuesta de una capa depende de la capa anterior, no de toda la red, lo cual se puede intuir en la forma en la que están conectadas capa a capa, ya que solo se conectan a una pequeña área de la capa anterior, en vez de conectar a toda la capa[68].

En esta capa, al igual que la convolucional, maneja una ecuación para la salida de las neuronas en la misma. A diferencia de la convolucional, además de su operación característica, los *feature maps* previos a esta capa son unidimensionales, es decir, las coordinadas 2D son ahora omitidas. De esta manera, la salida de cada neurona o nodo es computada por medio de la siguiente ecuación:

$$y_j = f\left(\sum_i w_{ij}\, x_i + b_j\right)$$

Entrenamiento: La CNN es una red *feedforward*, lo que infiere en que el entrenamiento de esta red sea supervisado, en donde la entrada, en el caso de la última etapa de la red, son las características extraídas de la previa etapa de extracción de características (*feature extraction*), y la salida, la cual son un conjunto de etiquetas.

Como se ha visto en las anteriores figuras, la primera etapa de extracción de características se compone de mapas, lo que quiere decir que las neuronas están organizadas en capas de 2D. De esta manera, las CNNs requieren un gran número de conexiones. Sin embargo, estas redes evitan el gran problema que puede ocasionar esto (un gran requerimiento computacional, difícil convergencia, etc.), al utilizar el principio de compartir los pesos sinápticos y los campos receptivos locales[69], disminuyendo la cantidad de parámetros de entrenamiento en

[67] WAGNER, Op. cit. p. 2
[68] MRAZOVA, Op. cit. p. 2.
[69] Ibíd.

la red. Esta es una de las grandes diferencias entre estas redes y una *feedforward* "normal".

Los campos receptivos locales (*local receptive field*) son otra característica o diferencia notable en una CNN, junto con el *pooling*. Los campos receptivos son el conjunto de nodos (cantidad de neuronas conectadas) en la capa de entrada de una capa que afectan la activación de una neurona. Se puede hacer una analogía al decir que el *receptive field* es, ya sea en las primeras capas, una imagen; o al final de la etapa de extracción de características, lo que la neurona puede abstraer o "ver"[70].

Ahora bien, el algoritmo de entrenamiento estándar de una CNN es el *backpropagation*[71,72,73]. Sin embargo, se hacen algunas modificaciones para incorporar los pesos sinápticos compartidos, que a su vez, involucran al tipo de red de la última etapa. Una vez cargados los pesos iniciales dándoles valores aleatorios y calcular las salidas; si en la última etapa se encuentran capas de neuronas RBF, la cual es convencional, los pesos antes de esta etapa, los cuales vienen de la última capa de *pooling* o *subsampling,* se ajustan a los valores tomados de las características y no se les permite adaptarse durante el entrenamiento[74].

Dicho vector de pesos cumple el rol de volverse el vector de etiquetas o *targets*, para la primera capa de RBFs. De este modo, la última capa de red tiene una salida se denota por:

$$y_j = \sum_{n=1}^{N} (y_i - w_{ij})^2$$

En donde w_{ij} corresponde a los pesos entre la neurona j y la neurona i, desde la capa anterior j. N es el número de neuronas en la capa. Ahora, al tener definida la salida para la red RBF para el entrenamiento por *backpropagation*, se define la función de error, para la cual se definen los patrones p para cada clase.

---

[70] WU, Op. cit. p. 12.
[71] SCHERER Dominik. Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors, Op. cit. p. 4.
[72] XIE, Op. cit. p. 12.
[73] MRAZOVA, Op. cit. p. 3.
[74] Ibíd.

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{n=1}^{N}\left(y_{j,p} - d_{j,p}\right)^2$$

Donde $y_{j,p}$ representa el valor acutal de la salida j para la entrada de muestra p y $d_{j,p}$ es el valor deseado para la clasificación de las clases de características.

Con las anteriores consideraciones, se aplica el algoritmo de *backpropagation*, el cual se explicó anteriormente.

Como se puede ha podido observar, la red posee varios parámetros y/o posibles diseños que se pueden elegir para implementar una CNN, como por ejemplo, la cantidad de capas convolucionales y *subsampling*, como también la cantidad de capas ocultas en la etapa de clasificación. Otro parámetro a elegir también es la función de activación de cada capa; en la siguiente tabla se muestra una recomendación para obtener mejores resultados según[75].

Tabla 2. Funciones de activación recomendadas para una CNN

Tipo de capa	**Función de activación**
Convoluvional	Identidad
Max-pooling	Rectificado lineal (*rectified linear*)
Fully connected (capa oculta)	Rectificado lineal (*rectified linear*)
Fully connected (capa de salida)	Tangente hiperbólica (para el criterio de MSE) *Softmax* (para el criterio de entropía cruzada)

[75] XIE, Op. cit. p. 12.

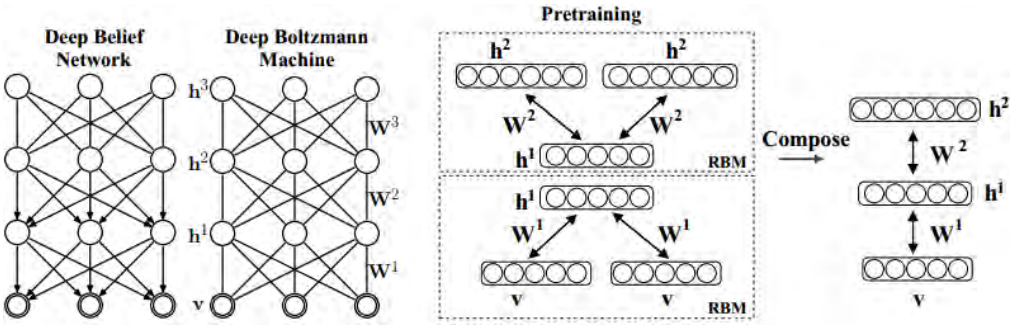
4.4. OTRAS ARQUITECTURAS

En las anteriores arquitecturas son los conceptos de *deep learning* más convencionales y representativos, por lo que se realiza una breve descripción de otras arquitecturas.

4.4.1. Deep Boltzmann Machine (DBM). Como su antecesora, la DBM es muy similar a la DBN. La DBM fue realizada por varias razones, como tener el potencial para aprender representaciones que se vuelven cada vez más complejas; como también construir una gran cantidad representaciones de alto nivel a partir de una gran cantidad de estímulos no etiquetados y limitados que pueden ser usados para afinar el modelo para una tarea específica[76].

Sin embargo, a diferencia de las DBNs, el entrenamiento de las DBMs le permite propagar mejor el error o la incertidumbre, ya que se incorpora un comportamiento en el que la aproximación del procedimiento de inferencia o muestras se pueden propagar de arriba hacia abajo y viceversa, como muestra la siguiente imagen.

Figura 17. Diferencia entre una DBN y DBM



Fuente: SALAKHUTDINOV, Ruslan; HINTON, Geoffrey E. Deep boltzmann machines [en línea]. En International Conference on Artificial Intelligence and Statistics. 2009. p. 4. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS09_SalakhutdinovH.pdf

[76] SALAKHUTDINOV, Ruslan; HINTON, Geoffrey E. Deep boltzmann machines. En International Conference on Artificial Intelligence and Statistics. 2009. p. 448-455. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS09_SalakhutdinovH.pdf

De esta manera, el pre-entrenamiento (*pretraining*) le permite a la red tener mejores resultados bajo ciertas condiciones[77]. Al igual que la DBN, los cuales son modelos de entrenamiento no supervisado[78], a la DBM también se pude lograr un fino ajuste del modelo[79], utilizando *backpropagation*.

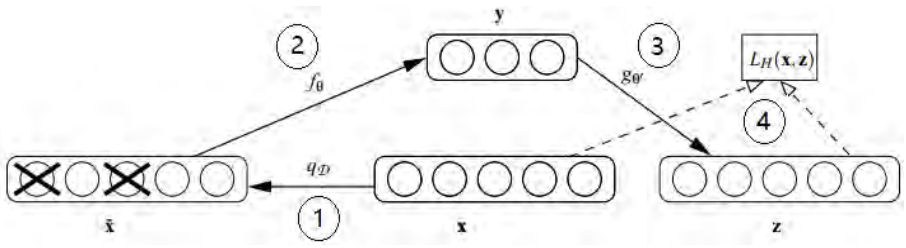
4.4.2. Stacked (denoising) Autoencoder (SDA). Un autoencoder realiza una reconstrucción por sí mismo de la entrada, pero la extracción de características que realiza es incapaz de garantizar características útiles, ya que puede conducir a una solución obvia (simplemente copiar la entrada). Para ello, se propuso un objetivo: limpiar la entrada que se encuentra "dañada" o *denoising*. Al lograrlo, se cumple un criterio de una buena representación ("una buena representación es una que puede ser obtenida desde una entrada dañada y que puede ser útil para recobrar la entrada original")[80]. Se enfatiza que el objetivo no es del todo el *denoising*, sino que tomar a este último como un criterio de entrenamiento para extraer útiles características.

De esta manera, se desarrolló un algoritmo para cumplir el objetivo de *denoising autoencoder*. En la siguiente ilustración se muestra los pasos de entrenamiento:

- **Paso 1:** "Corromper" la entrada x a $\tilde{x}$ por medio de un mapeo estocástico $\tilde{x} \sim q_D(\tilde{x}|x)$
- **Paso 2:** $\tilde{x}$ es mapeado en representaciones ocultas $y = f_\theta(\tilde{x}) = s(W\tilde{x} + b)$
- **Paso 3:** Desde y se reconstruye, $z = g_{\theta'}(y)$
- **Paso 4:** Entrenar parámetros para minimizar la entropía cruzada* "reconstrucción del error" [51] $L_{\mathrm{H}}(x, z) = \mathrm{H}(B_x || B_z)$, donde B_x denota la distribución multivariable de Bernoulli con parámetro x.

[77] Ibíd., p. 4.
[78] LI Deng. DONG Yu. Op. cit. p. 218.
[79] HINTON Geoffrey. A better way to pretrain deep Boltzmann machines, Op. cit. 7.
[80] VINCENT, Pascal, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion [en línea]. The Journal of Machine Learning Research, 2010, vol. 11, p. 3371-3408. [Consultado 06 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.3484&rep=rep1&type=pdf
* Al igual que el gradiente descendente, la entropía cruzada es otro criterio para encontrar el error mínimo.

Figura 18. Arquitectura y algoritmo de un denoising autoencoder

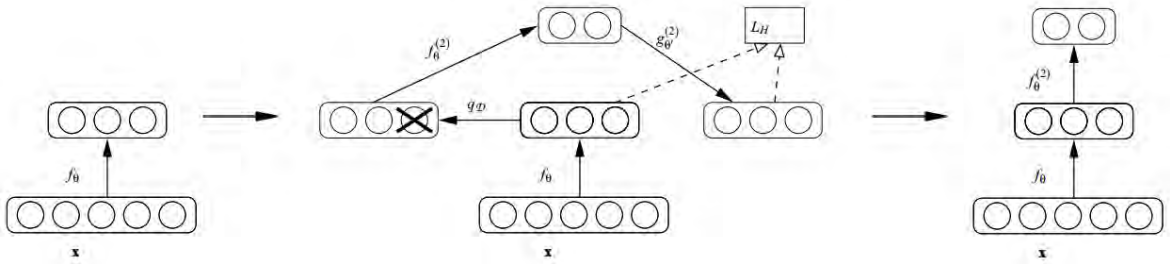


Fuente: VINCENT, Pascal, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion [en línea]. The Journal of Machine Learning Research, 2010, vol. 11, p. 9. [Consultado 06 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.3484&rep=rep1&type=pdf

Como se observa en la anterior figura, se puede apreciar una forma o arquitectura de *autoencoder*, a la cual se le ha aplicado el concepto de *denoising*. Ahora bien, se apilan varias de estas capas (*stacked*), como se muestra en la siguiente figura, aplicando el siguiente algoritmo.

- **Paso 1:** Mapear f_θ para aprender la formación de *denoising autoencoder*
- **Paso 2:** Usar a f_θ directamente como entrada para el siguiente nivel.
- **Paso 3:** Mapear y entrenar $f_\theta^{(2)}$ para el presente nivel
- **Paso 4:** Iterar para inicializar las capas posteriores

Figura 19. Apilamiento (stacking) de denoising autoencoder



Fuente: VINCENT, Pascal, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion [en línea]. The Journal of Machine Learning Research, 2010, vol. 11, p. 13. [Consultado 06 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.3484&rep=rep1&type=pdf

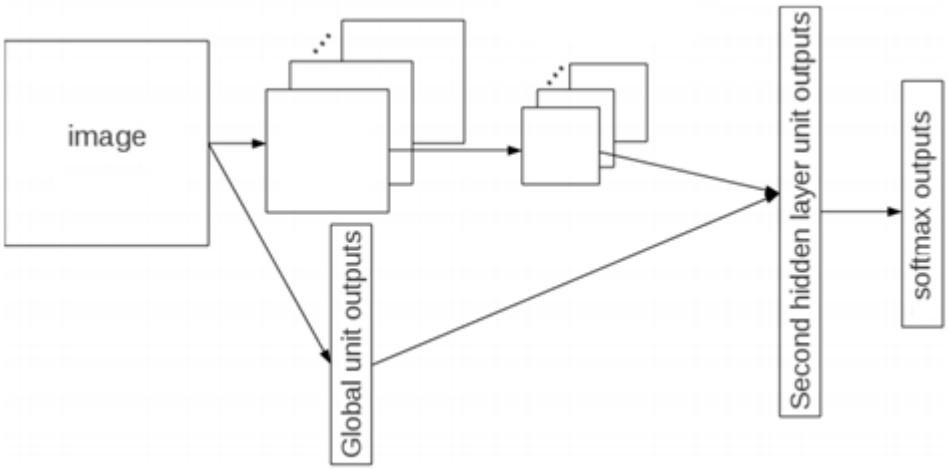
Al final, también se puede aplicar un ajusto fino, por medio de un entrenamiento supervisado[81].

4.4.3. Convolutional Deep Belief Networks (CDBN). Como su nombre lo indica, esta red se conforma por dos arquitecturas de *deep learning*. La arquitectura de una CDBN es muy similar a la de una CNN, como se muestra en la siguiente figura. La red presenta un pre-entrenamiento, en el cual la red CNN presente en la arquitectura se entrena, presentándole las imágenes o datos de entrada; una vez se ha realizo esto, la parte de la DBN (últimas dos capas) se entrenan con capas o unidades Gaussian-Bernoulli, las cuales, al no ser una red del todo supervisada, es muy útil este tipo de elección para clasificar las características[82].

[81] Ibíd., p. 12.

[82] KRIZHEVSKY, Alex; HINTON, G. Convolutional deep belief networks on cifar-10 [en línea]. Unpublished manuscript, 2010. p. 5. [Consultado 12 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.5826&rep=rep1&type=pdf

Figura 20. Arquitectura de una CDBN



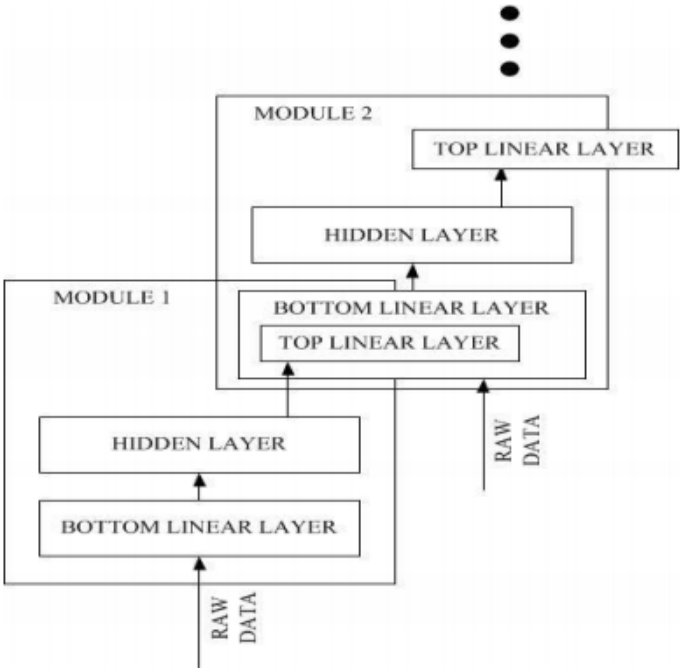
Fuente: KRIZHEVSKY, Alex; HINTON, G. Convolutional deep belief networks on cifar-10 [en línea]. Unpublished manuscript, 2010. p. 5. [Consultado 12 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.5826&rep=rep1&type=pdf

4.4.4. Deep Stacking Networks (DSN). Esta red, también llamada *deep convex network* (DCN) debido a que presenta un problema en el aprendizaje de los pesos, el cual es un problema de *convex optimization*[83], es una red que contiene varios módulos, donde cada uno de ellos son especialmente redes neuronales de una capa oculta y dos conjuntos entrenables de pesos[84].

[83] DENG, Li; YU, Dong; PLATT, John. Scalable stacking and learning for building deep architectures [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012. p. 2133-2136. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://research-srv.microsoft.com/pubs/157586/DSN-ICASSP2012.pdf

[84] DENG, Li; YU, Dong. Deep convex net: A scalable architecture for speech pattern classification [en línea]. En Proceedings of the Interspeech. 2011. p. 5. [Consultado 10 de diciembre, 2014]. Disponible en Internet: http://www.msr-waypoint.com/pubs/152133/DeepConvexNetwork-Interspeech2011-pub.pdf

Figura 21. Diagrama de bloques de dos módulos en una DSN



Fuente: DENG, Li; YU, Dong. Deep convex net: A scalable architecture for speech pattern classification [en línea]. En Proceedings of the Interspeech. 2011. p. 2. [Consultado 10 de diciembre, 2014]. Disponible en Internet: http://www.msr-waypoint.com/pubs/152133/DeepConvexNetwork-Interspeech2011-pub.pdf

De esta manera, la capa de entrada de la capa más baja es una capa lineal con un conjunto de entadas, estas corresponden a una "entrada bruta" (*raw data*) del vector de entrada para entrenamiento, es decir, si los vectores de entrada N en los datos totales de entrenamiento $X = [x_1, \dots, x_i, \dots x_N]$, con cada vector $x_i = [x_{1i}, \dots, x_{ji}, \dots x_{Di}]^T$; entonces las unidades de entrada corresponden a los elementos en x_i de dimensión D.

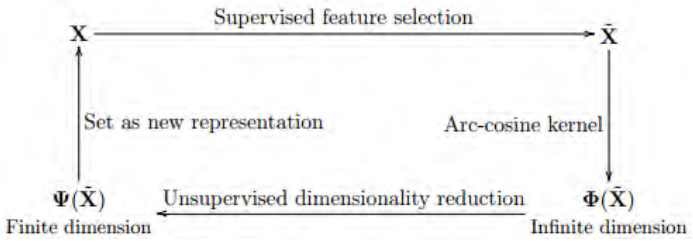
La capa oculta es, a modo de ejemplo, una capa que consiste de unidades no lineales, como una unidad sigmoidea. Donde la salida de esta se denota por $h_i = \sigma(W_1^T x_i)$, σ es la función sigmoidea y W_1 son los pesos entenables. La capa de salida consiste en conjunto de unidades lineales y cuya salida se denota por $y_i = U_1^T x_i$, donde U_1 son los pesos entrenables.

La salida de cada capa representan los objetivos (*targets*) de clasificación[85]. De una forma similar se van propagando los datos en el entrenamiento, en cual, como ya se mencionó, consiste en aplicar *convex optimization*.

Esta red posee un pre-entrenamiento, el cual se utiliza para inicializar los pesos del módulo inferior utilizando RBM[86].

4.4.5. Multilayer Kernel Machines (MKM). Este concepto surge de la idea de incorporar una reducción de dimensión no supervisado y las técnicas de selección de características supervisadas dentro de multicapas de *arc-cosine kernels*,*[87].

Figura 22. Concepto que sigue la MSM



Fuente: CHO, Youngmin. Kernel methods for deep learning [en línea]. Ph. D. Thesis. Department of Computer Science and Engineering. University of California, San Diego. 2012. p. 57. [Consultado 05 de diciembre, 2014]. Disponible en Internet: http://cseweb.ucsd.edu/~saul/papers/nips09_kernel.pdf

Como se muestra en la anterior ilustración, la entrada X, a la izquierda superior, es transformada o filtrada para que solo contenga las características relevantes de los datos etiquetados; el resultado $\tilde{X}$ es computado mediante *arc-cosine kernels,* los cuales implícitamente construyen una representación dimensional infinita $\Phi(\tilde{X})$. Seguidamente, se reduce la dimensión de $\Phi(\tilde{X})$ por medio de un método no supervisado, con el cual se obtiene a $\Psi(\tilde{X})$ y se lo fija como una entrada de características. Este ciclo se repite varias veces para formar una MKM.

---

[85] DENG Li. Scalable stacking and learning for building deep architectures,Op. cit. p. 1.
[86] Ibíd., p. 2.
[87] CHO, Youngmin. Kernel methods for deep learning [en línea]. Ph. D. Thesis. Department of Computer Science and Engineering. University of California, San Diego. 2012. p. 56. [Consultado 05 de diciembre, 2014]. Disponible en Internet: http://cseweb.ucsd.edu/~saul/papers/nips09_kernel.pdf
* arc-cosine kernels es una familia de kernels creada por el mismo autor de MSM, que imitan la computación de grandes RNA con una sola capa oculta.

Con esta red se demuestra que los métodos de *kernel* pueden ser ampliados para el concepto de *deep learning*, sin la necesidad de concetos complejos[88].

4.4.6. Spike-and-Slab RBMs (ssRBMs). El término *spike-and-slab* proviene de la literatura estadística[89] y se refiere a una mezcla entre dos componentes, *spike,* una masa de probabilidad discreta en cero; y la *slab*, una densidad (típicamente una distribución uniforme) sobre un dominio continuo. De forma análoga surge el concepto que trae esta red, ya que esta red, a diferencia de la RBM, contiene valores tanto reales (*slab*), como binarios (*spike*) asociados a variables con cada unidad en la capa oculta.

Ahora, con las consideraciones que esta red acapara, la capa visible posee Gaussian RMB, definidas como la distribución condicional de la capa visible dada la capa oculta forman una covarianza Gaussiana fija. Este tipo de consideración se utiliza para manejar los valores realizas mencionados. Obviamente, esto trae ciertos cambios en el algoritmo de entrenamiento, el cual, sin embargo, sigue teniendo como objetivo maximizar la verosimilitud.

4.4.7. Deep Predictive Coding Network (DPCN). Esta red fue inspirada bajo la idea de un sistema dinámico capaz de ajustarse o adaptarse al contexto de los datos, es decir, analógicamente a un sistema dinámico que puede controlarse o adaptarse de una perturbación o una nueva entrada. La pieza principal de este modelo es un procedimiento para inferir los estados dispersos (*sparse states*) de una red dinámica que se utiliza para la extracción de características[90].

La arquitectura de esta red se basa en que en cualquier capa, el sistema dinámico sea capaz de hacer la mejor predicción de la representación en la capa inferior, usando la información de las capas superiores o anteriores y la información temporal de los estados anteriores. De aquí el nombre de la red.

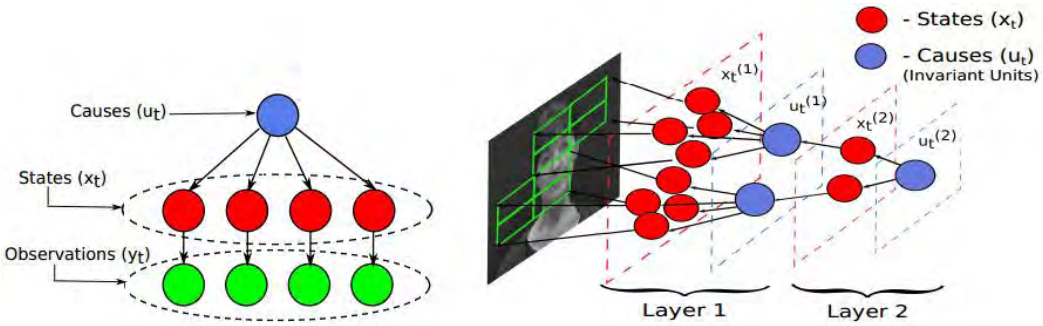
---

[88] Ibíd., p. 57

[89] BENGIO, Yoshua; COURVILLE, Aaron C.; BERGSTRA, James S. Unsupervised models of images by spike-and-slab RBMs [en línea]. En Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011. p. 1. Consultado 02 de diciembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Courville_591.pdf

[90] CHALASANI, Rakesh; PRINCIPE, Jose C. Deep predictive coding networks [en línea]. arXiv preprint arXiv:1301.3541, 2013. p. 1. [Consultado 03 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1301.3541.pdf

Figura 23. Arquitectura de una DPCN. Izquierda: Una capa de la DPCN, en la cual se muestra que la conforman unidades de observación (entradas, verdes), estados (rojos) y causa (azul). Derecha: Una DPCN con dos capas



Fuente: CHALASANI, Rakesh; PRINCIPE, Jose C. Deep predictive coding networks [en línea]. arXiv preprint arXiv:1301.3541, 2013. p. 3. [Consultado 03 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1301.3541.pdf

Como se muestra en la anterior figura, la arquitectura es muy enfocada en un sistema dinámico, pues posee en una capa, unidades de observación, estado y causa. Además, las capas están arregladas como un esquema *Markov chain*, de tal forma que los estados en cualquier capa son solo dependan de las representaciones en la capa inferior y superior, y son independientes del resto del modelo.

El entrenamiento de la red se realiza mediante un método o algoritmo no supervisado, el cual es *greedy layer-wise*. Se menciona que también es posible extender la red a una forma similar a la CNN[91].

[91] Ibíd., p. 2.

4.5. TAXONOMÍA

El concepto de *deep learning* puede clasificarse de varias formas, ya que como se ha mostrado, el mismo acapara muchos otros conceptos. Sin embargo, se puede destacar el tipo de entrenamiento de las arquitecturas, como también, el tipo de redes que la conforman, como se muestra en las siguientes ilustraciones.

Figura 24. Taxonomía del deep learning a partir de conceptos de RNA

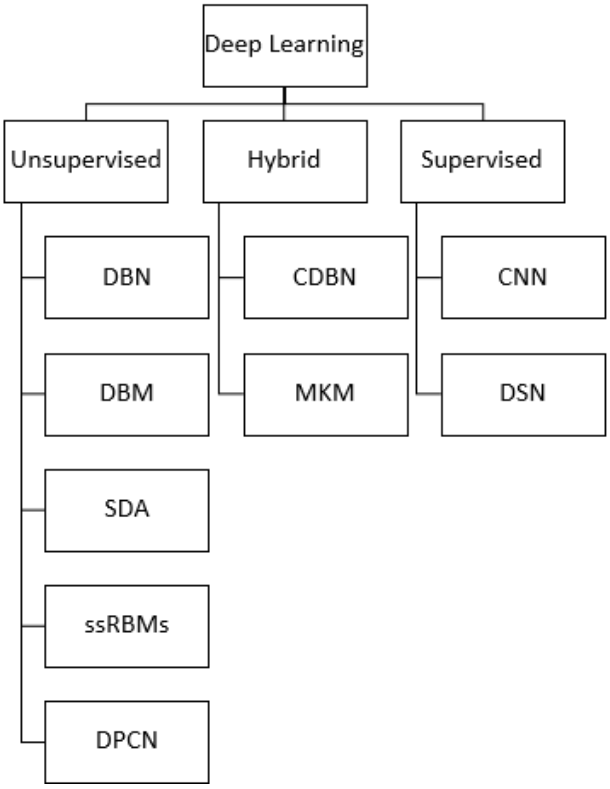
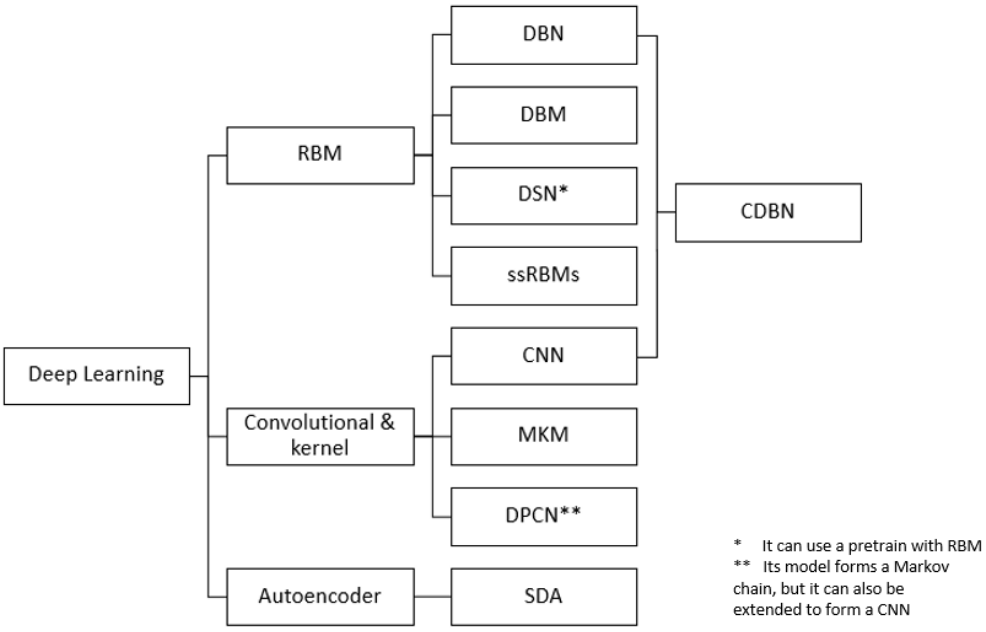


Figura 25. Taxonomía del deep learning a partir del tipo de algoritmo de aprendizaje



En la taxonomía a partir del algoritmo de aprendizaje, se clasifico las redes de acuerdo a su algoritmo más común, ya que de no ser así, la mayoría de las redes serian de algoritmos híbridos, pues pueden tener un pre-entrenamiento de un tipo diferente al entrenamiento formal. En cuanto a la taxonomía a partir de conceptos de RNA, se clasifican los conceptos de acuerdo a la arquitectura de RNA que implementan.

Al clasificar las diferentes arquitecturas, se puede notar que el énfasis, principalmente en las RBMs es bastante común, esto se debe a su gran nivel de abstracción, al igual que la operación de convolución. Estos dos conceptos acaparan las arquitecturas *deep learning*, incluso, llegando a una combinación de ambos conceptos para un mejor resultados, en la red CDBN.

5. REVISIÓN DE APLICACIONES DEL DEEP LEARNING

Ahora bien, además de los grandes avances mencionados anteriormente, los investigadores del tema en cuestión han realizado varias aplicaciones en las que se muestra el gran potencial del concepto de *deep learning*. A forma de ejemplo:

- **LeNet-5, reconocimiento de números.** Esta es una de las publicaciones[92] y aplicación[93] más conocida del *deep learning*. Esto se debe a que, podría afirmarse, aquí se consolido el concepto de la red *convolutional neural network* (CNN).

La red mencionada se implementa para clasificar, en principio, los diez primeros números (0 – 10). A pesar de que puede verse sencillo de hacer a primera vista, la red no necesita un procesamiento previo y es invariante a las transformaciones (rotación, translación, etc.) e incluso al ruido.

La red desarrollada consta de 7 capas, la entrada son imágenes de 32x32 pixeles. Se utilizó un conjunto de datos denominado MNIST con el cual fue entrenada la red. Este consta de 60000 imágenes de entrenamiento y 10000 de prueba.

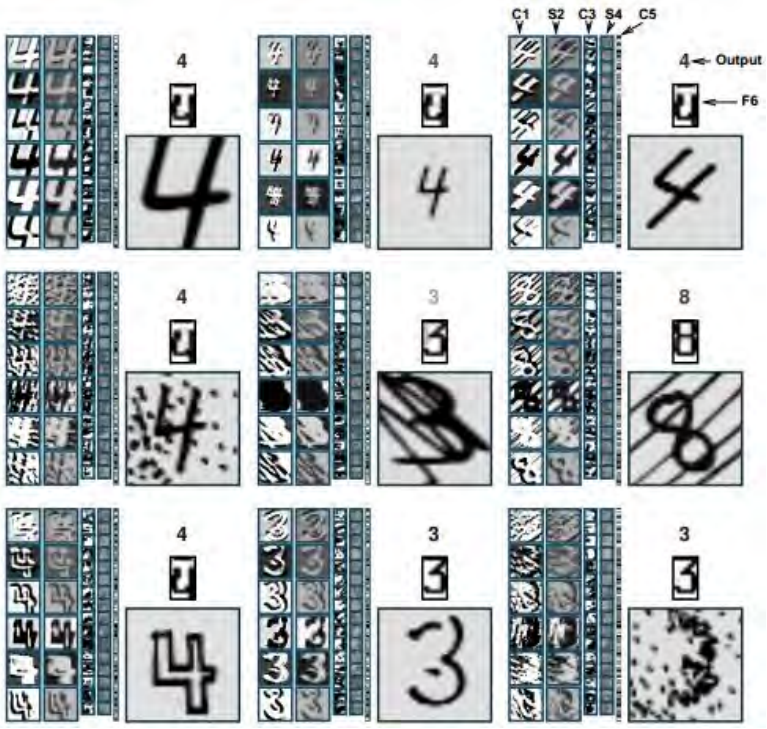
[92] LECUN, Yann. Gradient-based learning applied to document recognition, Op cit.
[93] LECUN, Yann. LeNet-5, convolutional neural networks [en línea]. 1998. [Consultado enero 11, 2015]. Disponible en Internet: http://yann.lecun.com/exdb/lenet/

Figura 26. Datos de entrenamiento, tomados de la MNIST para la CNN

Fuente: LECUN, Yann, et al. Gradient-based learning applied to document recognition [en línea]. Proceedings of the IEEE, 1998, vol. 86, no 11, p. 10. [Consultado 2 de octubre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

La red presento un buen desempeño, logrando una tasa error para los datos de prueba de 0.7%. El mejor de su tiempo (1998).

Figura 27. Pruebas realizadas con transformaciones y ruido en las imágenes de prueba



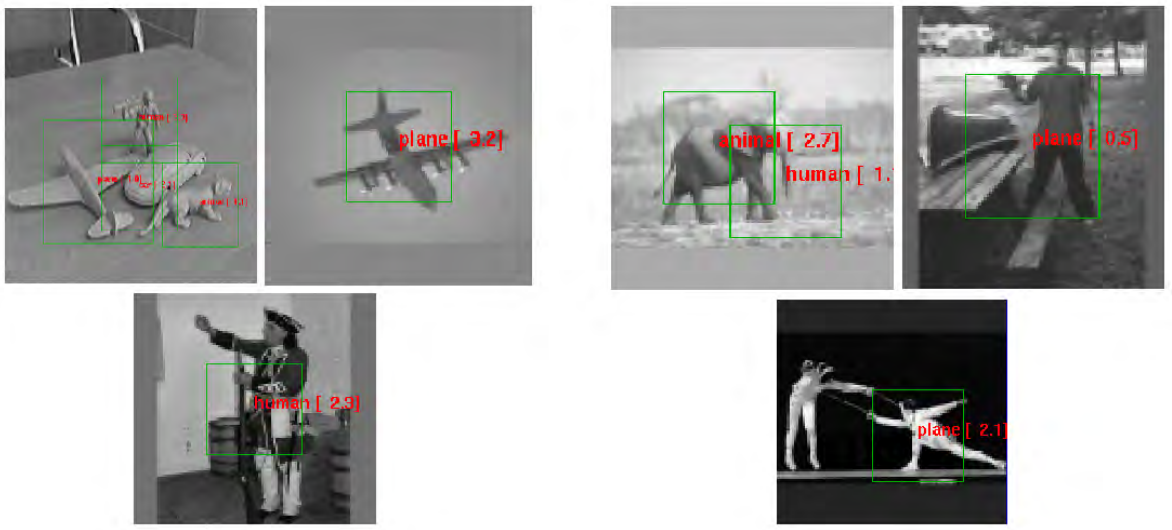
Fuente: LECUN, Yann, et al. Gradient-based learning applied to document recognition [en línea]. Proceedings of the IEEE, 1998, vol. 86, no 11, p. 17. [Consultado 2 de octubre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

- **Reconocimiento de objetos genéricos en imágenes.** En esta aplicación se entrena una red neuronal convolucional (CNN) para clasificar e identificar cinco categorías: Animales de cuatro patas, figuras humanas, aeroplanos, carros y camiones. Para lograr esta tarea, la red se entrenó con una base de datos de 24300 muestras y otras 24300, para la validación de la red[94].

El autor de esta aplicación obtuvo un error de 6.2% con su propia CNN, lo cual es un error admisible. En las siguientes figuras se muestran ejemplos de su aplicación:

[94] HUANG, F. J. LECUN, Y. BOTTOU, L. Generic Object Recognition in Images (NORB) [en línea]. 2003. [Consultado 19 de diciembre, 2014]. Disponible en Internet: http://www.cs.nyu.edu/~yann/research/norb/index.html

Figura 28. Resultados del reconocimiento de objetos. Izquierda: Resultados aceptados. Derecha: Resultados erroneous



Fuente: HUANG, F. J. LECUN, Y. BOTTOU, L. Generic Object Recognition in Images (NORB) [en línea]. 2003. [Consultado 19 de diciembre, 2014]. Disponible en Internet: http://www.cs.nyu.edu/~yann/research/norb/index.html

- **Extracción de características usando aprendizaje no supervisado.** Los autores y/o creadores de la aplicación se plantearon el problema de aprender y reconocer a detectar características específicas usando solo datos (imágenes) sin etiquetar. Ellos lo lograron, detectando o no detectando rostros de personas, y para ello entrenaron una red *sparse autoencoder* de 9 capas con un conjunto de datos de 10 millones imágenes de 200x200 pixeles[95].

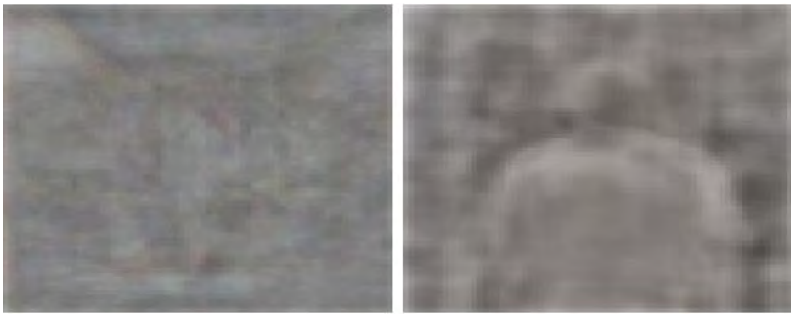
[95] LE, Quoc V. Building high-level features using large scale unsupervised learning [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013. p. 8595-8598. [Consultado 16 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1112.6209.pdf&embed

Figura 29. Algunas imágenes de entrenamiento

Fuente: LE, Quoc V. Building high-level features using large scale unsupervised learning [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013. p. 5. [Consultado 16 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1112.6209.pdf&embed

Para poner a prueba la red, observando que tenían muy buenos detectando rostros de personas, utilizaron varios rostros de la cara de diferentes gatos (al igual que los rostros de personas), con lo cual la red diferenciaba o no, si la imagen correspondía a un humano o un gato. Obtuvieron una mejora de mejora de hasta un 70% respecto a su previo estado del arte.

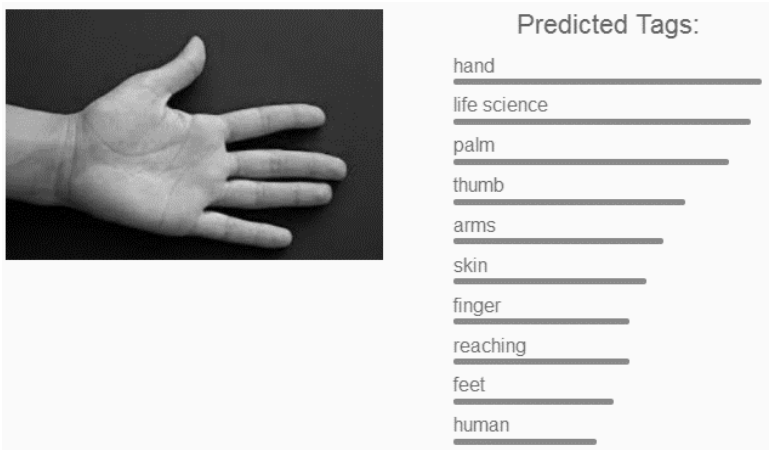
Figura 30. Visualización de la cara del gato (izquierda) y el cuerpo humano (derecha)



Fuente: LE, Quoc V. Building high-level features using large scale unsupervised learning [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013. p. 6. [Consultado 16 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1112.6209.pdf&embed

- **Clasificador de imágenes.** Esta es una aplicación web. Esta aplicación clasifica 1000 diferentes clases de objetos en una imagen. La red utilizada es una red neuronal convolucional (CNN), la cual fue entrenada con cerca de 1.3 millones de imágenes de muestras[96]. Esta aplicación puede identificar en una imagen varios objetos, como muestra en la siguiente figura:

Figura 31. Resultado de una imagen cargada en la aplicación



Fuente: Image Classifier Demo [en Línea]. New York University, 2013. [Consultado 27 de diciembre, 2014]. Disponible en Internet: http://horatio.cs.nyu.edu/

- **Etiquetado de escenas.** Un paso importante hacia la "comprensión" de una imagen o escena es realizar un etiquetado de la misma, esta técnica es denominada *scene parsing* y consiste en etiquetar cada pixel en una imagen con una categoría del objeto a la que este perteneciera[97]. Eso fue lo que se buscaba en este proyecto, desarrollar un sistema que "entendiera" o identificara los objetos presentes en una escena y los etiquetara en la misma.

Para ello, implementaron un sistema del cual hace parte una red *convolutional neural network*. A pesar de ciertos inconvenientes que se presentaron por el

[96] Image Classifier Demo [en Línea]. New York University, 2013. [Consultado 27 de diciembre, 2014]. Disponible en Internet: http://horatio.cs.nyu.edu/

[97] FARABET, Clement, et al. Learning hierarchical features for scene labeling [en línea]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, vol. 35, no 8, p. 10-12. [Consultado 27 de diciembre, 2014]. Disponible en Internet: https://hal-upec-upem.archives-ouvertes.fr/hal-00742077/document

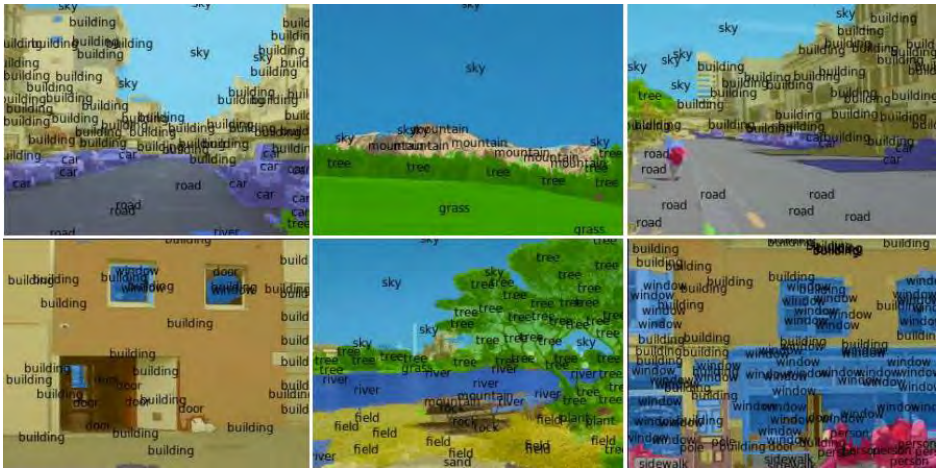
etiquetado de cada pixel, implementaron una red denominada *multi-scale convolutional network,* con la cual eliminarían dicho inconveniente.

Los resultados que obtuvieron con su sistema fueron impresionantes, ya que batieron en precisión y velocidad a los 3 sistemas de etiquetado antecedentes, mejorando en velocidad en un factor de 100.

Figura 32. Resultados obtenidos de la red implementada

Fuente: FARABET, Clement, et al. Learning hierarchical features for scene labeling [en línea]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, vol. 35, no 8, p. 10. [Consultado 27 de diciembre, 2014]. Disponible en Internet: https://hal-upec-upem.archives-ouvertes.fr/hal-00742077/document

Figura 33. Resultados obtenidos en el conjunto de datos de uno de los antecedentes



Fuente: FARABET, Clement, et al. Learning hierarchical features for scene labeling [en línea]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, vol. 35, no 8, p. 12. [Consultado 27 de diciembre, 2014]. Disponible en Internet: https://hal-upec-upem.archives-ouvertes.fr/hal-00742077/document

- **Detección de rostros.** Para esta aplicación desarrollaron un novedoso método que, simultáneamente, detecta y estima la posición y orientación de los rostros (*pose estimation*) en tiempo real. El sistema se basa en una red CNN para convertir las imágenes en puntos sobre un colector de baja dimensión para determinar la *pose estimation*[98].

La CNN utilizada tiene una arquitectura muy similar a la de LeNet-5, pero contiene más *feature maps.* Posee la misma entrada de 32x32 pixeles en escala de grises. El sistema fue entrenado con 52850 imágenes de entrenamiento (rostros) del tamaño de la entrada, en diferentes posiciones (diferentes ángulos del rostro y posiciones del mismo), como también con el mismo número de imágenes de entrenamiento para etiquetar áreas en donde no existía la ausencia del rostro.

Los autores demuestran que un sistema entrenado que detecta simultáneamente el rostro y *pose estimation* es más preciso que las dos tareas realizadas por sistemas separados; obteniendo los siguientes resultados.

[98] OSADCHY, Margarita; CUN, Yann Le; MILLER, Matthew L. Synergistic face detection and pose estimation with energy-based models. *The Journal of Machine Learning Research*, 2007, vol. 8, p. 1197-1215.

Figura 34. Resultados obtenidos por el sistema de detección de rostros

Fuente: OSADCHY, Margarita; CUN, Yann Le; MILLER, Matthew L. Synergistic face detection and pose estimation with energy-based models [en línea]. The Journal of Machine Learning Research, 2007, vol. 8, p. 15. [Consultado 10 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/osadchy-07.pdf

- **LARG: Learning Applied to Ground Robotics.** Ahora, el concepto de *deep learning* es naturalmente aplicado al campo de la robótica. El propósito de este proyecto era diseñar algoritmos de aprendizaje para permitir a un robot navegar en entornos complejos por medio de cámaras[99]. Este proyecto realizado entre varias entidades posee varias tecnologías aplicadas, y entre ellas algunas redes del

[99] LECUN, Yann. LARG: Learning Applied to Ground Robotics [en línea]. 2004. [Consultado 12 de enero, 2015]. Disponible en Internet: http://www.cs.nyu.edu/~yann/research/lagr/

presente concepto; algunas son *deep belief network* (DBN) y *convolutional neural network* (CNN).

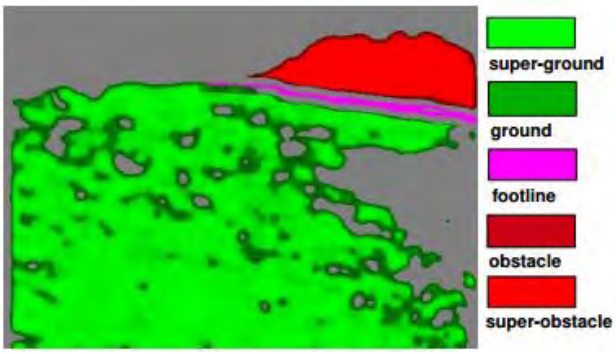
Se utilizaron los principios de la *deep belief network* y el procesamiento de una CNN. La arquitectura de la red consta de 3 capas apiladas y fue entrenada con 150 diversos escenarios con un total de 10000 imágenes de diferentes resoluciones[100]. De esta manera, se etiquetaron 5 clases de escenarios, obteniendo los siguientes resultados.

Figura 35. Etiquetas para el entrenamiento de la red de LARG

Fuente: LECUN, Yann, et al. Deep belief net learning in a long-range vision system for autonomous off-road driving [en línea]. En Intelligent Robots and systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008. p. 5. [Consultado 20 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/hadsell-iros-08.pdf

[100] LECUN, Yann, et al. Deep belief net learning in a long-range vision system for autonomous off-road driving [en línea]. En Intelligent Robots and systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008. p. 628-633. [Consultado 20 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/hadsell-iros-08.pdf

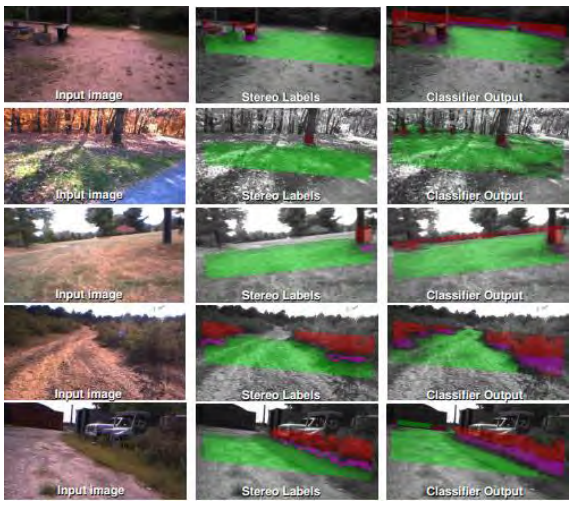
Figura 36. 5 etiquetas aplicadas en una escena para la aplicación



Fuente: LECUN, Yann, et al. Deep belief net learning in a long-range vision system for autonomous off-road driving [en línea]. En Intelligent Robots and systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008. p. 5. [Consultado 20 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/hadsell-iros-08.pdf

Cabe destacar que las redes mencionadas hacen parte de un sistema de visión, de esta manera se obtuvieron los siguiente resultados.

Figura 37. Pruebas de campo del sistema de visión



Fuente: LECUN, Yann, et al. Deep belief net learning in a long-range vision system for autonomous off-road driving [en línea]. En Intelligent Robots and systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008. p. 6. [Consultado 20 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/hadsell-iros-08.pdf

- **Automatización fenotípica del desarrollo de embriones.** Como su nombre deducir, el *deep learning* se adentra en el campo de la biología y genética. A pesar de entrar en este campo se aplica el mismo concepto, el uso de imágenes, pero en este caso, se hace uso del video. En este proyecto se elaboró un sistema entrenable para analizar videos del desarrollo de embriones *C. elegans*.

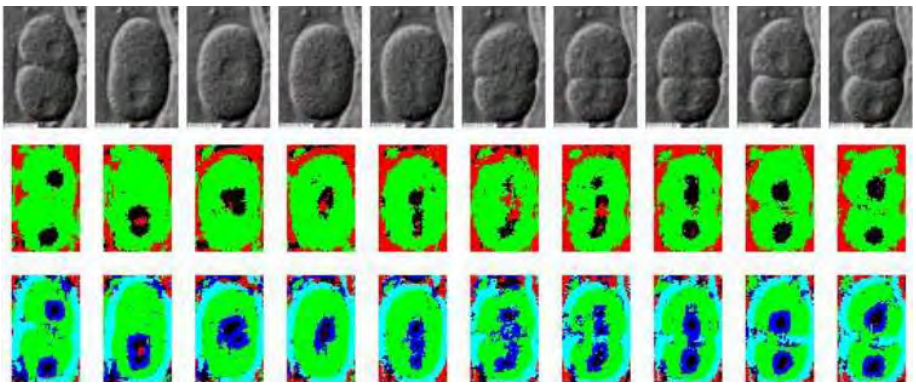
El sistema automáticamente detecta, segmenta, y localiza las células y su núcleo en las imágenes del microscopio. El sistema consta de tres módulos, entre ellos una red CNN, la cual clasifica cada pixel en 5 categorías: Pared celular, citoplasma, membrana nuclear, núcleo y extracelular[101].

La CNN implementada consta de 6 capas y posee una entrada de 40x40 pixeles. Las imágenes de entrenamiento fueron extraídas de 5 diferentes películas de embriones *C. elegans,* de las cuales extrajeron 50 *frames*. Para las imágenes de prueba se extrajeron 30 *frames* de 3 diferentes películas. Sin embargo, cada *frame* tenía un tamaño de 300x300 pixeles, por lo cual, tomaron la decisión de recortar los *frames* en imágenes de 40x40 pixeles, obteniendo un total de 190440 imágenes en total (las cuales estaban distribuidas para las 5 categorias) para el entrenamiento.

Cumpliendo con el objetivo, obtuvieron buenos resultados como se muestra en la siguiente figura, en el cual se diferencias las 5 etiquetas por sus diferentes colores.

[101] NING, Feng, et al. Toward automatic phenotyping of developing embryos from videos [en línea]. Image Processing, IEEE Transactions on, 2005, vol. 14, no 9, p. 1360-1371. [Consultado 11 de diciembre, 2014]. Disponible en Internet: https://hal.archives-ouvertes.fr/hal-00114920/document

Figura 38. Resultados obtenidos para el proyecto del desarrollo de embriones C. elegans



Fuente: NING, Feng, et al. Toward automatic phenotyping of developing embryos from videos [en línea]. Image Processing, IEEE Transactions on, 2005, vol. 14, no 9, p. 6. [Consultado 11 de diciembre, 2014]. Disponible en Internet: https://hal.archives-ouvertes.fr/hal-00114920/document

6. EJEMPLO DE APLICACIÓN

Como se ha podido observar anteriormente, el *deep learning* puede acaparar un gran número de aplicaciones, y para cada una de ellas es necesario un rico conjunto de datos (*datasets*). La diversidad y calidad de dichos datos, tanto de entrenamiento como de prueba, son uno de los parámetros más fundamentales de la red. Sin dichos datos, la arquitectura, tipo de red, cantidad de capas ocultas, entre otros parámetros, no tendrían tanta importancia.

En este caso, se hizo uso de un *dataset* comúnmente utilizado en el campo de *machine learning*. Este campo ha implementado varias técnicas de aprendizaje con dicho conjunto, ya que es una buena base para nuevas técnicas, como lo menciona Yann LeCun[102], una de las importantes figuras del campo, tanto de *machine learning,* como también de *deep learning*.

El *dataset* utilizado es MNIST (*Mixed National Institute of Standards and Technology*), es un conjunto de números o dígitos escritos a mano. Consiste de pequeñas imágenes (28x28 pixeles) en escala de grises, y contiene 60000 datos de entrenamiento y 10000 de prueba. Este conjunto de datos es el mismo utilizado en la aplicación LeNet-5.

Para la aplicación se utilizó el *toolbox* desarrollado por Rasmus Berg Palm[103] para MATLAB. Dicho *toolbox* contiene cuatro arquitecturas de *deep learning*, las cuales son: *deep belief network* (DBN), *convolutional neural network* (CNN), *stacked autoencoder* (SAE) y *convolutional autoencoder* (CAE). A pesar de que esta última red no se explicó anteriormente, de acuerdo a las demás redes explicadas se puede intuir como es su estructura o arquitectura.

Las arquitecturas empleadas para la aplicación fueron la DBN y CNN. En cada una se entrena la red con los datos de entrenamiento, y se la simula con los datos de prueba, es decir, se realiza un ejercicio de clasificación. Con dichos datos se evalúa la respuesta y el error de cada red. También se introduce ruido a los datos de prueba para evaluar la generalización de cada red, al igual que su respectivo error.

---

[102] LECUN, Yann; CORTES, Corinna; BURGES, Christopher. THE MNIST DATABASE of handwritten digits [en línea]. [Consultado 13 de enero, 2015]. Disponible en Internet: http://yann.lecun.com/exdb/mnist/

[103] PALM, Rasmus Berg. Deep Learning toolbox [en línea]. Mayo 2014. [Consultado 13 de enero, 2015]. Disponible en Internet: http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox

- **MNIST con Deep Belief Network.** Como bien se sabe, una DBN es una arquitectura que por defecto no es supervisada; este tipo de entrenamiento es muy útil para detectar o extraer las características de los datos. Por ello, es necesario introducir un entrenamiento supervisado el cual se encarga de clasificar dichas características.

En la parte no supervisada, se carga el conjunto de datos MNIST teniendo en cuenta que estos datos tienen un formato *uint8*, es decir, la imagen se encuentra en escala de grises. Sin embargo, una red DBN admite datos binarios o normalizados en un rango de 0 a 1. Por esta razón, se normaliza las imágenes y se las convierte en formato *double*.

Ahora bien, los objetivos, salidas deseadas o *targets*, no se ingresan de la misma forma que al utilizar el *toolbox* de RNA de MATLAB. Por el contrario, cada vector de *targets* de cada característica o dato de entrada, va en un vector fila, no en un vector columna, como en el *toolbox* de MATLAB.

Para configurar la arquitectura de la DBN, se crean dos "vectores estructura" (*structure array*). Uno de ellos es un vector fila el cual lleva la cantidad de neuronas de cada capa, es decir, cada columna del mencionado vector denota una capa, y el valor en cada columna, la cantidad de neuronas. El segundo y último vector estructura tiene 4 parámetros, número de iteraciones, número de muestras para el entrenamiento, *momentum* y *alpha* o razón de aprendizaje. Se puede decir que el primer vector estructura hace referencia a la estructura, mientras que el segundo hace alusión a los parámetros de entrenamiento. Se muestra un ejemplo en la siguiente tabla.

Tabla 3. Descripción de los parámetros de la arquitectura y de entrenamiento de la DBN

Descripción	Sintaxis	Rango	Ejemplo
Número de capas y neuronas	name1.size	Enteros positivos	dbn.sizes = [100 100]
Número de iteraciones	name2.numepochs	Enteros positivos	opts.numepochs = 3
Número de muestras	name2.batchsize	Enteros positivos	opts.batchsize = 100
Valor *momentum*	name2.momentun	0-1	opts.momentum = 0.1
Valor *alpha* o razón de aprendizaje	name2.alpha	0-1	opts.alpha = 1

No obstante, además de las consideraciones como el rango, la red posee otra consideración, la cual es que la relación entre la cantidad de datos de entrenamiento y el parámetro *batchsize,* la cual debe ser entera. Esta consideración es realizada por el autor del *toolbox* y se podría decir que lo que busca esta consideración es darle una división uniforme a los datos.

Finalmente, la red la arquitectura es creada mediante el comando *dbnsetup* y entrenada con la instrucción *dbntrain*, a las cuales se les ingresan en este orden: el vector estructura de la arquitectura, los datos de entrenamiento y el vector de estructura del entrenamiento.

Al obtener los pesos de la DBN, estos son pasados y/o desplegados en una RNA, que el autor del *toolbox* solamente nombra por *neural network* (NN), la cual contiene las mismas capas de la DBN y cada capa tiene los respectivos pesos de la capa que tenía en DBN. Además, se agrega una última capa, la cual posee las neuronas que el diseñador requiera para etiquetar las características de esta etapa supervisada.

Al igual que la DBN, se crea un vector estructura en el cual se ingresan la cantidad de iteraciones y el número de muestras para entrenar esta red supervisada. Una vez se eligieron los datos, se entrena la red con el comando *nntrain.* La instrucción utilizada en esta red para simular y evaluar los datos de prueba es *nntest*, a la cual se le ingresan: la red entrenada, los datos de prueba y sus respectivas salidas esperadas o *targets*.

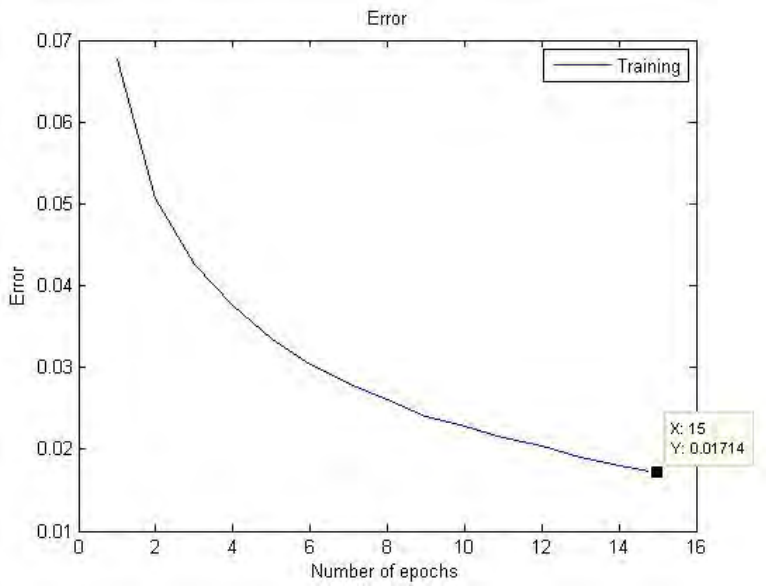
Ahora bien, se calibró y/o ajusto la red completa (parte no supervisada y no supervisada) de tal forma que la tasa de error para los datos de prueba fuera menor que el 10%, el cual es el error admisible para el creador del *toolbox*. De esta forma, los valores tomados para la red fueron:

Tabla 4. Valores tomados para calibrar y/o ajustar a la DBN para un error menor a 10%

Entrenamiento	Parámetro	Valor
No supervisado (DBN)	Número de capas y neuronas	[784 300]
	Número de iteraciones	4
	Número de muestras	200
	Valor *momentum*	0
	Valor *alpha* o razón de aprendizaje	0.5
Supervisado (NN)	Número de iteraciones	15
	Número de muestras	200

Con los valores registrados en la anterior tabla, se obtuvo una tasa de error para los datos de prueba de MNIST de 2.52%, lo cual está dentro del rango acertado por el creador del *toolbox*. La tasa de error durante el entrenamiento para 15 iteraciones de la NN fue de 1.71%, como se muestra en la siguiente figura. Esto quiere decir que la red posee una generalización destacable, pues posee un error muy bajo y les es muy fácil detectar los dígitos con esta configuración o entrenamiento realizado.

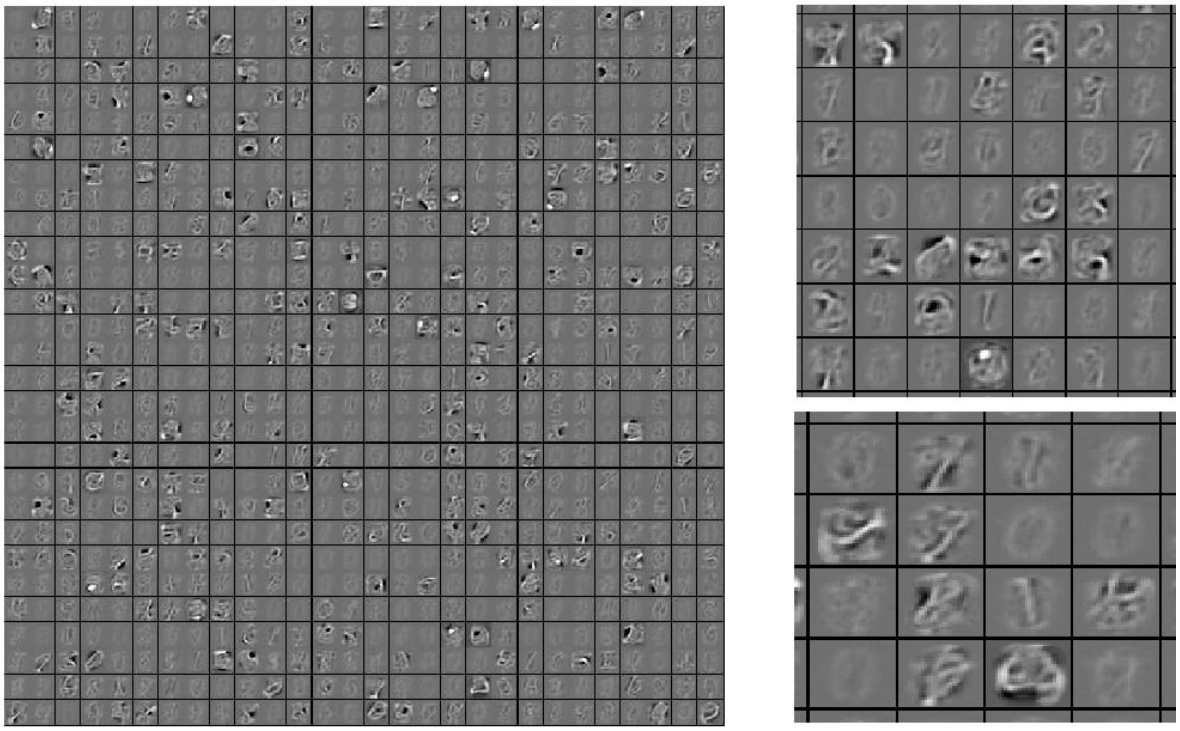
Figura 39. Tasa de error de entrenamiento para la DBN



Los pesos de cada neurona de cada capa en la DBN, se pueden visualizar por medio de una función del *toolbox* denominada *visualize*. Esta función acomoda los

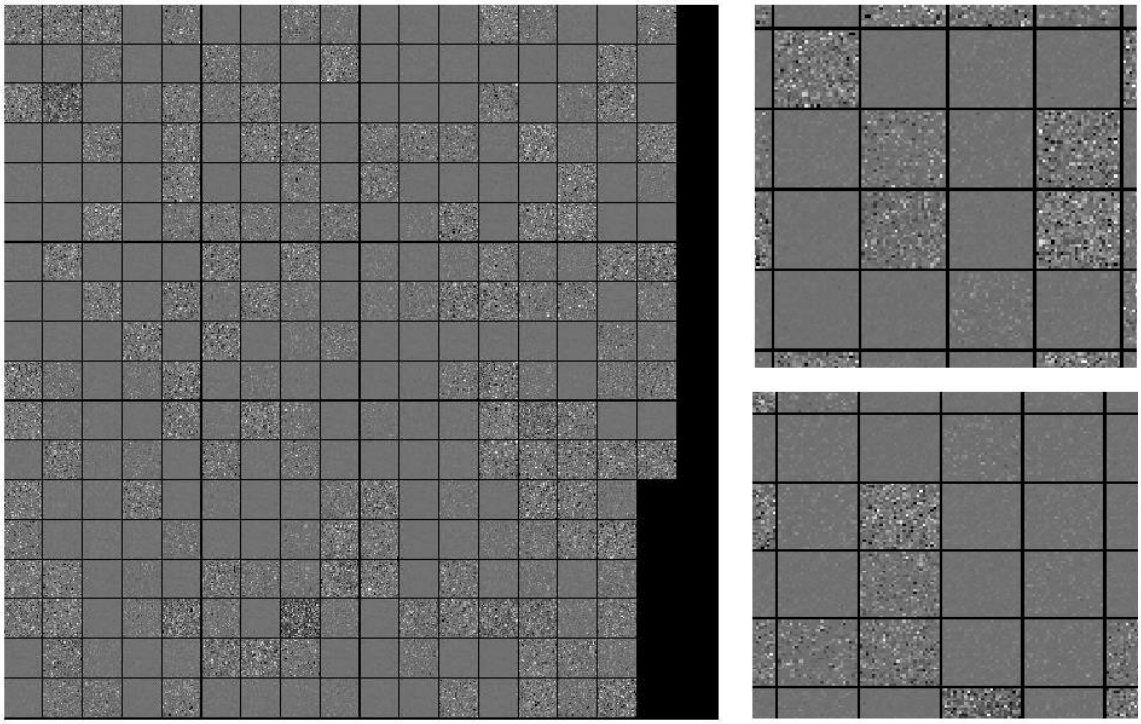
pesos de forma cuadrada, dándole la forma de una imagen del mismo tamaño, tanto de filas como de columnas.

Figura 40. Pesos de la primera capa de la DBN. Izquierda: Todos los pesos de la primera capa. Derecha: Vista más detallada



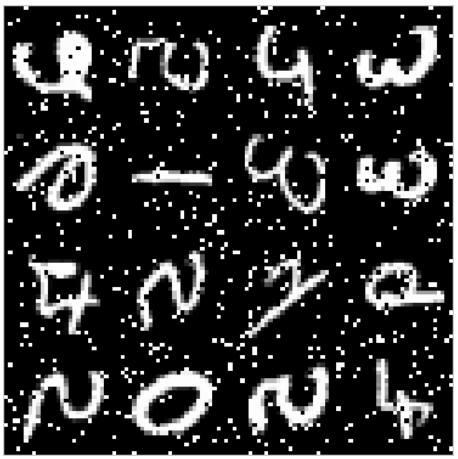
Como se puede observar en la anterior figura, cada recuadro es una neurona, y en cada una de ellas se puede ver que se solapan los números, de tal manera que se difuminan. En la siguiente capa se puede observar que lograr entender la imagen de cada neurona no es sencillo, de hecho, se puede decir que las características, a medida que se siguen abstrayendo de capa en capa, se van volviendo más abstractas.

Figura 41. Pesos de la segunda capa de la DBN. Izquierda: Todos los pesos de la segunda capa. Derecha: Vista más detallada



Con ello, se probó adicionándole ruido a los datos de prueba. El ruido aplicado fue '*salt & pepper*' con una densidad de 0.1. De esta manera, se obtuvieron imágenes de prueba como las de a continuación:

Figura 42. Ruido aplicado a los datos de prueba de MNIST



Para estos datos de prueba, se obtuvo un error de 8.97%, por lo cual se sigue cumpliendo con la recomendación del autor y se puede afirmar que la red fue bien entrenada, pues posee, como ya se mencionó, una buena generalización.

- **MNIST con Convolutional Neural Network.** Como la arquitectura de esta red es totalmente supervisada; no posee un pre-entrenamiento para detectar las características de los datos. A pesar de no contar con un entrenamiento no supervisado, posee entre sus capas iniciales dos operaciones que le permiten extraer dichas características y las cuales también le brindan un grado de invariancia a la clasificación que realiza la red.

Al igual que en la DBN, el *dataset* MNIST es cargado teniendo en cuenta los requerimientos de los datos de entrada. Estos son básicamente dos. El primero es, al igual que en la anterior arquitectura, los datos deben estar en un rango de 0 a 1. El segundo y último son las dimensiones de los datos de entrada, ya que esta debe ser una matriz tridimensional, es decir, la red al ser especialmente diseñada para el procesamiento de imágenes, admite datos bidimensionales, y cuya tercer dimensión indica la cantidad de patrones que la red debe aprender. En este caso, la red posee una entrada de 28x28x60000 imágenes.

Las salidas deseadas o *targets* de esta red se organizan diferente a la DBN, es decir, igual que con el *toolbox* de RNA de MATLAB, en vector columna para cada respectivo patrón de entrenamiento. Ahora bien, la arquitectura de la red es muy similar a la DBN; requiere de dos *structure array*, uno para los datos o parámetros de la arquitectura y otro para los parámetros de entrenamiento.

El vector estructura de la arquitectura requiere del ingreso de varios parámetros: la especificación de una capa de entrada, la cantidad de mapas de características y el tamaño o dimensión del *kernel* para la capa de convolución y, para la capa de *subsampling* se requiere del tamaño de la escala. De igual forma que la DBN, el vector estructura de entrenamiento requiere el ingreso de la razón de aprendizaje (*alpha*), el número de iteraciones y el tamaño de la muestra de los datos de entrada.

Tabla 5. Parámetros del structure array para la arquitectura

Tipo de capa	Parámetros	Sintaxis	Ejemplo
Entrada	Tipo se capa	*struct('type', 'i')*	Igual que la sintaxis
Convolucional	Tipo de capa. Número de mapas de características. Tamaño del *kernel.*	struct('type', 'c', 'outputmaps', #, 'kernelsize', #)	struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5)
Subsampling	Tipo de capa. Tamaño de la escala.	struct('type', 's', 'scale', #)	struct('type', 's', 'scale', 3)

A pesar que los valores y/o parámetros de entrenamiento no tiene alguna condición para la ejecución del algoritmo, la arquitectura de la red debe ser planteada de acuerdo a unos criterios. El diseño de la arquitectura es muy dependiente del tamaño de cada mapa de características; por esta razón, se debe tener en cuenta el tamaño de la capa anterior. Para el *toolbox* en cuestión, la cantidad de *feature maps* de la capa de convolución es la misma que la de *subsampling*. No obstante, el criterio más destacable es el valor de la escala en la capa de *subsampling*. Este valor, de acuerdo al *toolbox*, "escala" los mapas de características por medio de la relación entre el tamaño de los mismos y el valor dado al parámetro *scale*. Esta relación debe ser entera.

$$subsamplingsize(t) = \frac{subsamplingsize(t-1)}{scale}$$

Para diseñar o determinar el tamaño de cada mapa de una capa convolucional, se utiliza la siguiente expresión, la cual depende del tamaño del mapa anterior y del tamaño del *kernel* en cuestión.

$$mapsize(t) = mapsize(t-1) - kernelsize(t) + 1$$

Con las anteriores expresiones o ecuaciones se puede determinar el tamaño de cada mapa de características de cada capa; teniendo en cuenta que la relación que con el parámetro de la capa de *subsampling*.

Para el entrenamiento, primero se debe crear la arquitectura por medio del comando *cnnsetup*, al cual se le ingresan el vector estructura de la arquitectura, los datos de entrenamiento y sus respectivas salidas deseadas o *targets*. El entrenamiento es realizado por medio de la instrucción *cnntrain*, a la cual se le

ingresan los mismos datos que al comando anterior, adicionando al final el vector estructura de entrenamiento.

Finalmente, se ajustó la red con los siguientes valores para obtener una tasa de error menor a 12% (recomendada por el creador del *toolbox*):

Tabla 6. Valores tomados para calibrar y/o ajustar a la CNN para un error menor a 12%

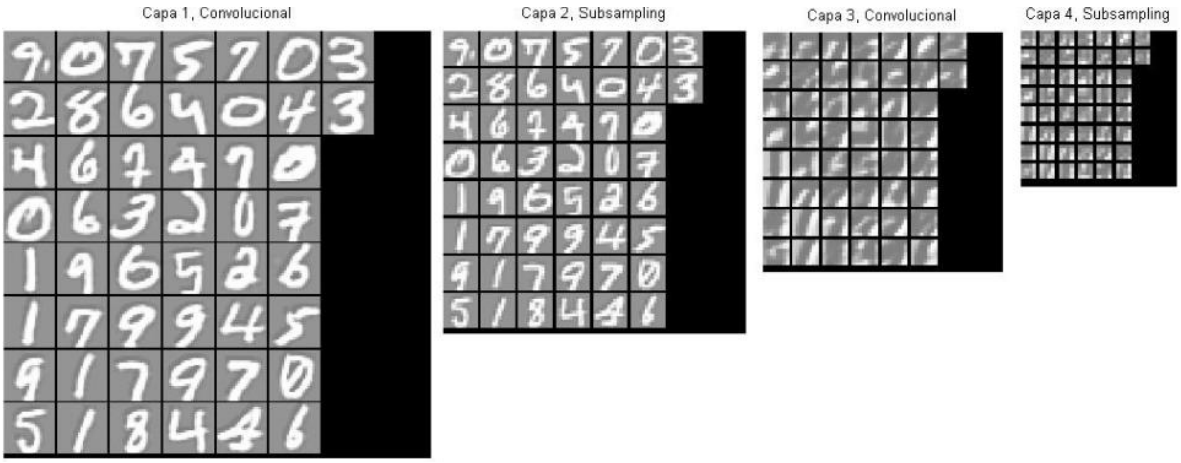
Capa	**Parámetro**	**Valor**
Convolucional	Número de mapas	6
	Tamaño del *kernel*	5 (5x5)
Subsampling	Escala (*scale*)	2
Convolucional	Número de mapas	12
	Tamaño del *kernel*	5 (5x5)
Subsampling	Escala (*scale*)	2
Todas	Razón de aprendizaje (*alpha*)	1
	Número de muestras	50
	Número de iteraciones	10

Con los valores de la tabla anterior, se obtuvo una tasa de error del 2.73% para los datos de prueba de MNIST. Al igual que la DBN, para los valores suministrados posee una buena generalización. Para comprobarlo, se realizó la misma prueba que en la DBN, al agregar ruido a las muestras de MNIST. Con ellas se obtuvo un error de 9.6%, lo cual sigue estando dentro del rango recomendado por el autor del *toolbox.*

Al igual que en la DBN, se extrajeron las características de cada capa, en este caso, se visualizan solo los resultados de las operaciones (convolución o *subsampling*) para las 50 muestras elegidas de un solo mapa de características.

Cada recuadro es un mapa de características para una determinada entrada (50 en total). Cada recuadro de la capa no. 1 (convolucional) posee un tamaño de 24x24 pixeles. Los recuadros de la capa no. 2 (*subsampling*) poseen un tamaño de 12x12 pixeles cada uno, mientras que los de la capa no. 3 poseen un tamaño de 8x8 pixeles, y en la última capa (*subsampling*), cada recuadro posee un tamaño de 4x4 pixeles.

Figura 43. Pesos de la CNN implementada



Como se puede observar, a medida que aumenta el número de capas, las características son más abstractas y los mapas más pequeños. Cabe recordar que solo se muestra uno de los mapas de características para las 50 muestras tomadas para entrenar la red.

Ahora bien, para las aplicaciones anteriores se utilizó un computador con procesador Intel core i7-2620 y RAM de 6.0 GB. Con este equipo cada algoritmo de entrenamiento, DBN y CNN, se demoraron 518.2 y 1530.3s respectivamente, es decir, para la DBN se tomó 8.63 minutos, mientras que para la CNN se tomó 25.5 minutos.

De acuerdo a los datos registrados, la red DBN es mejor que la CNN. Sin embargo, se debe tener en cuenta muchos aspectos antes de poder afirmar del todo que una arquitectura es mejor que otra, ya que, como se ha podido observar, una red tiene parámetros que la otra no, como también una tiene dos tipos de aprendizaje, etc. Por lo que es muy complejo afirma cual red en definitiva es la mejor.

En los anexos se muestran los *scripts* o códigos para cada red.

7. CONCLUSIONES Y TRABAJO FUTURO

El *deep learning* es un concepto que inculca el uso de la jerarquía en sus arquitecturas o redes. Este es el concepto que, a consideración del autor, se ha inculcado y se puede ver claramente en cada una de las redes que conforman dicho concepto. Para lograr ese objetivo, diferentes autores se han apropiado de las mejores RNA del campo de *machine learning*, y como se ha mostrado, estas redes son un su mayoría RBMs y *Autoencoders*, además de utilizar otros conceptos como el de la convolución.

Cabe destacar que es fundamental el uso de los diferentes tipos de aprendizajes, tanto no supervisado, para extraer características, como supervisado, para poder clasificar dichas características. De esta manera, el *deep learning* también ha implementado esos tipos de aprendizajes formando redes hibridas, las cuales poseen un *pretrain* o pre-entrenamiento que les brindan mejores propiedades y resultados que una sola red. Este es el caso de la DBN, la cual es fundamentalmente no supervisada. Sin embargo, se le introduce una parte supervisada para clasificar las características, brindando buenos resultados, mejores que los de una red que solo posee un solo tipo de entrenamiento, como lo es la CNN. No obstante, es muy prematuro afirmar que una red es mejor que otra, ya que todo puede depender del diseño y de la experiencia del usuario para entrenar dichas redes, como de otras consideraciones.

Finalmente, se ha podido observar la aplicabilidad del *deep learning* en diferentes campos del conocimiento, desde la caligrafía o escritura a mano, hasta la genética. Todas estas aplicaciones se han realizado por medio del procesamiento de imágenes, datos bidimensionales, por lo que también es viable utilizar datos unidimensionales, como el sonido o audio.

Con la aplicación elaborada en este trabajo se busca motivar y ensañar los parámetros que conforman este concepto, sin olvidar de visualizar las características extraídas en las cuales se denota el *deep learning*. También se puede observar el gran trabajo computacional, además de matemático, que requiere una arquitectura.

El trabajo futuro va encaminado en el desarrollo y un estudio más profundo de las arquitecturas que cumplen con el concepto en cuestión, además de estudiar formas de optimización de código, para implementar más arquitecturas que tratan de “imitar” el comportamiento del celebro y que las grandes industrias tienen en la mira.

BIBLIOGRAFÍA

AREL, Itamar; ROSE, Derek C.; KARNOWSKI, Thomas P. Deep machine learning-a new frontier in artificial intelligence research [en línea]. Computational Intelligence Magazine, IEEE, 2010, vol. 5, no 4, p. 13-18. [Consultado 07 de octubre, 2014]. Disponible en Internet: http://msrds.googlecode.com/svn/trunk/rbm/DML_Arel_2010.pdf

Artificial intelligence scientist gest $1M prize [en línea]. CBC News, 2011. [Consultado 25 de marzo, 2014]. Disponible en Internet: http://www.cbc.ca/news/technology/artificial-intelligence-scientist-gets-1m-prize-1.1034093#ixzz1DxUEvAcQ

BENGIO, Yoshua; COURVILLE, Aaron C.; BERGSTRA, James S. Unsupervised models of images by spike-and-slab RBMs [en línea]. En Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011. p. 1145-1152. [Consultado 02 de diciembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Courville_591.pdf

BENGIO, Yoshua, et al. Greedy layer-wise training of deep networks [en línea]. Advances in neural information processing systems, 2007, vol. 19, p. 153. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2006_739.pdf

BENGIO, Yoshua. Learning deep architectures for AI [en línea]. Foundations and trends® in Machine Learning, 2009, vol. 2, no 1, p. 1-127. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf

BENBIO, Yoshua. Learning deep architectures for AI [en línea]. Technical Report 1312, Université de Montréal, 2007. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf

BOUVRIE, Jake. Notes on convolutional neural networks [en línea]. Center for Biological and Computer Learning, Department of Brain and Cognitive Sciences. MIT, Cambridge, 2006. [Consultado 03 de octubre, 2014]. Disponible en Internet: http://cogprints.org/5869/1/cnn_tutorial.pdf

CHALASANI, Rakesh; PRINCIPE, Jose C. Deep predictive coding networks [en línea]. arXiv preprint arXiv:1301.3541, 2013. [Consultado 03 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1301.3541.pdf

CHO, Youngmin. Kernel methods for deep learning [en línea]. Ph. D. Thesis. Department of Computer Science and Engineering. University of California, San Diego, 2012. 118 p. [Consultado 05 de diciembre, 2014]. Disponible en Internet: http://cseweb.ucsd.edu/~saul/papers/nips09_kernel.pdf

CIRESAN, Dan C., et al. Flexible, high performance convolutional neural networks for image classification [en línea]. En IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No.1. 2011. p. 1237. [Consultado 12 de octubre, 2014]. Disponible en Internet: http://people.idsia.ch/~juergen/ijcai2011.pdf

CLAUDIU CIRESAN, Dan, et al. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition [en línea]. arXiv preprint arXiv:1003.0358, 2010. [Consultado 01 de abril, 2014]. Disponible en Internet: http://arxiv.org/pdf/1003.0358.pdf

CONG, Jason; XIAO, Bingjun. Minimizing Computation in Convolutional Neural Networks [en línea]. En Artificial Neural Networks and Machine Learning–ICANN 2014. Springer International Publishing, 2014. p. 281-290. [Consultado 10 de octubre, 2014]. Disponible en Internet: http://vast.cs.ucla.edu/sites/default/files/publications/CNN_ICANN14.pdf

DENG, Li; YU, Dong. Deep convex net: A scalable architecture for speech pattern classification [en línea]. En Proceedings of the Interspeech. 2011. [Consultado 10 de diciembre, 2014]. Disponible en Internet: http://www.msr-waypoint.com/pubs/152133/DeepConvexNetwork-Interspeech2011-pub.pdf

DENG, Li; YU, Dong; PLATT, John. Scalable stacking and learning for building deep architectures [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012. p. 2133-2136. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://research-srv.microsoft.com/pubs/157586/DSN-ICASSP2012.pdf

GULCEHRE, Caglar. Deep Learning – Bibliography [en línea]. Septiembre 2014. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://deeplearning.net/bibliography/

DESJARDINS, Guillaume. Training deep convolutional architectures for vision [en línea]. Master Tesis. Departamento de informática e Investigación Operativa, Facultad de Artes y Ciencias, 2009. 116 p. [Consultado 20 de septiembre, 2014]. Disponible en Internet:https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/3646/Desjardins_Guillaume_2009_memoire.pdf?sequence=2

FARABET, Clement, et al. Learning hierarchical features for scene labeling [en línea]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, vol. 35, no 8, p. 1915-1929. [Consultado 27 de diciembre, 2014]. Disponible en Internet: https://hal-upec-upem.archives-ouvertes.fr/hal-00742077/document

FISCHER, Asja; IGEL, Christian. An introduction to restricted Boltzmann machines [en línea]. En Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Berlin Heidelberg, 2012. p. 14-36. [Consultado 21 de septiembre, 2014]. Disponible en Internet: http://image.diku.dk/igel/paper/AItRBM-proof.pdf

GRAFF, P. FEROZ, F. HOBSON, M. P. LASENBY, A. SKYNET: An efficient and robust neural network training tool for machine learning in astronomy [en línea]. Monthly Notices of the Royal Astronomical Society 441 (2014). p. 1741-1759. [Consultado 15 de septiembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1309.0790v2.pdf

HERNANDEZ, Daniela. Meet the Man Google Hired to Make AI a Reality [en línea]. Enero 2014. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://www.wired.com/wiredenterprise/2014/01/geoffrey-hinton-deep-learning

HINTON, Geoffrey E.; SALAKHUTDINOV, Ruslan R. A better way to pretrain deep Boltzmann machines [en línea]. En Advances in Neural Information Processing Systems. 2012. p. 2447-2455. [Consultado 11 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~fritz/absps/DBM_pretrain.pdf

HINTON, Geoffrey; OSINDERO, Simon; TEH, Yee-Whye. A fast learning algorithm for deep belief nets [en línea]. Neural computation, 2006, vol. 18, no 7, p. 1527-1554. http://www.cs.toronto.edu/~hinton/absps/fastnc.pdf

HUANG, F. J. LECUN, Y. BOTTOU, L. Generic Object Recognition in Images (NORB) [en línea]. 2003. [Consultado 19 de diciembre, 2014]. Disponible en Internet: http://www.cs.nyu.edu/~yann/research/norb/index.html

HUBEL, David H.; WIESEL, Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [en línea]. The Journal of physiology, 1962, vol. 160, no 1, p. 106-154. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1962.sp006837/pdf

Image Classifier Demo [en Línea]. New York University, 2013. [Consultado 27 de diciembre, 2014]. Disponible en Internet: http://horatio.cs.nyu.edu/

KOLLER, Daphne; FRIEDMAN, Nir. Probabilistic graphical models: principles and techniques [en línea]. MIT press, 2009. [Consultado 15 de octubre, 2014]. Disponible en Internet: http://mitpress.mit.edu/sites/default/files/titles/content/9780262013192_sch_0001.pdf

KRIZHEVSKY, A. Learning multiple layers of features from tiny images [en línea]. Master's thesis, Computer Science Department, University of Toronto, 2009. 60 p. [Consultado 11 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

KRIZHEVSKY, Alex; HINTON, G. Unpublished manuscript, Convolutional deep belief networks on cifar-10 [en línea]. 2010. [Consultado 12 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.5826&rep=rep1&type=pdf

LE, Quoc V. Building high-level features using large scale unsupervised learning [en línea]. En Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE

International Conference on. IEEE, 2013. p. 8595-8598. [Consultado 16 de diciembre, 2014]. Disponible en Internet: http://arxiv.org/pdf/1112.6209.pdf&embed
LECUN, Yann; CORTES, Corinna; BURGES, Christopher. THE MNIST DATABASE of handwritten digits [en línea]. [Consultado 13 de enero, 2015]. Disponible en Internet: http://yann.lecun.com/exdb/mnist/

LECUN, Yann, et al. Deep belief net learning in a long-range vision system for autonomous off-road driving [en línea]. En Intelligent Robots and systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008. p. 628-633. [Consultado 20 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/hadsell-iros-08.pdf

LECUN, Yann, et al. Gradient-based learning applied to document recognition [en línea]. Proceedings of the IEEE, 1998, vol. 86, no 11, p. 2278-2324. [Consultado 2 de octubre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

LECUN, Yann; HUANG, Fu Jie; BOTTOU, Leon. Learning methods for generic object recognition with invariance to pose and lighting [en línea]. En Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, 2004. p. II-97-104. [Consultado 28 de septiembre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1315150&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1315150

LECUN, Yann. LARG: Learning Applied to Ground Robotics [en línea]. 2004. [Consultado 12 de enero, 2015]. Disponible en Internet: http://www.cs.nyu.edu/~yann/research/lagr/

LECUN, Yann. LeNet-5, convolutional neural networks [en línea]. 1998. [Consultado enero 11, 2015]. Disponible en Internet: http://yann.lecun.com/exdb/lenet/

LEE, Honglak, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations [en línea]. En Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009. p. 609-616. [Consultado 05 de octubre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.802&rep=rep1&type=pdf

LI, Deng. DONG, Yu. Deep Learning: Methods and Applications [en línea]. Foundations and Trends® in Signal Processing. Now Publish. 2014. [Consultado 20 de septiembre, 2014]. Disponible en Internet: http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf

LI, Yue. Review of Boltzmann Machine and simulated annealing [en línea]. CSC321 Tutorial 9. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://www.cs.utoronto.ca/~yueli/CSC321_UTM_2014_files/tut9.pdf

LOPES, Noel; RIBEIRO, Bernardete; GONÇALVES, Joao. Restricted Boltzmann machines and deep belief networks on multi-core processors [en línea]. En Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE, 2012. p. 1-7. [Consultado 13 de octubre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6252431&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6252431

LÓPEZ S, Jesús A. CAICEDO B, Eduardo F. “Una aproximación práctica a las redes neuronales artificiales”. Editorial Universidad Del Valle, 2009. 217 p.

MRAZOVA, Iveta; KUKACKA, Marek. Hybrid convolutional neural networks [en línea]. En Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. IEEE, 2008. p. 469-474. [Consultado 30 de septiembre, 2014]. Disponible en Internet:http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4618146&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4618146

NG, Andrew. Sparse Autoencoder [en línea]. CS294A Lecture notes, 72 (2011). [Consultado 10 de septiembre, 2014]. Disponible en Internet: http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

NING, Feng, et al. Toward automatic phenotyping of developing embryos from videos [en línea]. Image Processing, IEEE Transactions on, 2005, vol. 14, no 9, p. 1360-1371. [Consultado 11 de diciembre, 2014]. Disponible en Internet: https://hal.archives-ouvertes.fr/hal-00114920/document

OSADCHY, Margarita; CUN, Yann Le; MILLER, Matthew L. Synergistic face detection and pose estimation with energy-based models [en línea]. The Journal of Machine Learning Research, 2007, vol. 8, p. 1197-1215. [Consultado 10 de diciembre, 2014]. Disponible en Internet: http://yann.lecun.com/exdb/publis/pdf/osadchy-07.pdf

PALM, Rasmus Berg. Deep Learning toolbox [en línea]. Mayo 2014. [Consultado 13 de enero, 2015]. Disponible en Internet: http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox

Restricted Boltzmann Machine – Short Tutorial [en línea]. iMONAD. [Consultado 13 de enero, 2015]. Disponible en Internet: http://imonad.com/rbm/restricted-boltzmann-machine/

TORRES SOLER, Luis Carlos. Redes Neuronales [en línea]. Diciembre 2010. [Consultado 11 de abril, 2014]. Disponible en Internet: http://disi.unal.edu.co/~lctorress/RedNeu/LiRna008.pdf

SALAKHUTDINOV, Ruslan. Deep Learning, KDD Tutorial [en línea]. Department of Computer Science. Department of Statistics. University of Toronto. Canadian Institute for Advanced Research (CIFAR). 2014. [Consultado 20 de octubre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/talk_KDD_part1_pdf.pdf

SALAKHUTDINOV, Ruslan. Learning Deep Generative Models [en línea]. Ph. D. Thesis. Department of Electrical and Computer Engineering, University of Toronto. 2009. 84 p. [Consultado 24 de septiembre, 2014]. Disponible en Internet: http://www.cs.toronto.edu/~rsalakhu/papers/Russ_thesis.pdf

SALAKHUTDINOV, Ruslan; HINTON, Geoffrey E. Deep Boltzmann machines [en línea]. En International Conference on Artificial Intelligence and Statistics. 2009. p. 448-455. [Consultado 30 de noviembre, 2014]. Disponible en Internet: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS09_SalakhutdinovH.pdf

SCHERER, Dominik; MÜLLER, Andreas; BEHNKE, Sven. Evaluation of pooling operations in convolutional architectures for object recognition [en línea]. En Artificial Neural Networks–ICANN 2010. Springer Berlin Heidelberg, 2010. p. 92-101. [Consultado 02 de octubre, 2014]. Disponible en Internet: http://www.ais.uni-bonn.de/papers/icann2010_maxpool.pdf

SCHERER, Dominik; SCHULZ, Hannes; BEHNKE, Sven. Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors [en línea]. En Artificial Neural Networks–ICANN 2010. Springer Berlin Heidelberg, 2010. p. 82-91. [Consultado 29 de septiembre, 2014]. Disponible en Internet: http://www.is.uni-bonn.de/papers/icann10_conv.pdf

SCHMIDHUBER, Jürgen. "My First Deep Learning System of 1991 + Deep Learning Timeline 1962-2013" [en línea]. Diciembre 2013. [Consultado 28 de marzo, 2014]. Disponible en Internet: http://www.idsia.ch/~juergen/firstdeeplearner.html

SIMARD, Patrice Y.; STEINKRAUS, Dave; PLATT, John C. Best practices for convolutional neural networks applied to visual document analysis [en línea]. En 2013 12th International Conference on Document Analysis and Recognition. IEEE Computer Society, 2003. p. 958-958. [Consultado 28 de septiembre, 2014]. Disponible en Internet: http://vnlab.ce.sharif.ir/courses/85-86/2/ce667/resources/root/15%20-%20Convolutional%20N.%20N./ICDAR03.pdf

TIVIVE, Fok Hing Chi; BOUZERDOUM, Abdesselam. Efficient training algorithms for a class of shunting inhibitory convolutional neural networks [en línea]. Neural Networks, IEEE Transactions on, 2005, vol. 16, no 3, p. 541-556. [Consultado 21 de septiembre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1427760&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1427760

VINCENT, Pascal, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion [en línea]. The Journal of Machine Learning Research, 2010, vol. 11, p. 3371-3408. [Consultado 06 de diciembre, 2014]. Disponible en Internet: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.3484&rep=rep1&type=pdf

WAGNER, Raimar, et al. Learning convolutional neural networks from few samples [en línea]. En Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, 2013. p. 1-7. [Consultado 11 de octubre, 2014]. Disponible en Internet: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6706969&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6691896%2F6706705%2F06706969.pdf%3Farnumber%3D6706969

WU, David J. End-to-End Text Recognition with Convolutional Neural Networks [en línea]. Bachelor's Thesis with Honors and Distinction. Department of Computer Science. Stanford University, 2012. 60 p. [Consultado 07 de octubre, 2014]. Disponible en Internet: http://crypto.stanford.edu/~dwu4/papers/HonorThesis.pdf

XIE, Xiaohui; KIM, In-Jung. Handwritten Hangul recognition using deep convolutional neural networks [en línea]. International Journal on Document Analysis and Recognition (IJDAR), 2015, Vol. 18. Issue 1, p. 1-13. [Consultado 25 de septiembre, 2014]. Disponible en Internet: http://www.ics.uci.edu/~xhx/publications/HHR.pdf

ANEXOS

Anexo A. Script para la red DBN

```
%% Cargar y acondicionamiento los datos para la red
clear,clc

% Cargar el dataset MNIST
load mnist_uint8;

% Normalizacion de los datos
x_entrenamiento = double(train_x)/255;
x_prueba = double(test_x)/255;
y_entrenamiento = double(train_y);
y_prueba = double(test_y);

%%  Entrenamiento de la DBN (No supervisado)

% Parametros de la arquitectura

% cantidad de capas y numero de neuronas
dbn.sizes = [784 300];
% numero de iteraciones
opts.numepochs =   5;
% numero de muestras a tomar
opts.batchsize = 200;
opts.momentum  =   0;
% razon de aprendizaje
opts.alpha     =   0.5;
% se crea la arquitectura
dbn = dbnsetup(dbn, x_entrenamiento, opts);
% se entrena la red
dbn = dbntrain(dbn, x_entrenamiento, opts);

% se extraen los pesos de cada capa
mapas_capa1=dbn.rbm{1}.W';
mapas_capa2=dbn.rbm{2}.W';

% se visualiza los mapas
figure,visualize(mapas_capa1);
figure,visualize(mapas_capa2);

%% Entrenamiento supervisado (Red neuronal NN)

% pasa los pesos de la dbn a una red neuronal con 10 etiquetas o salidas
nn = dbnunfoldtonn(dbn, 10);
% funcion de activacion de la red neuronal sigmoidal
nn.activation_function = 'sigm';
```

```
% Parametros de la red neuronal NN

% cantidad de iteraciones
opts.numepochs =  15;
% numero de muestras de los datos de entrenamiento
opts.batchsize = 200;
% se habilita el grafico del error
opts.plot = 1;

% Entrenamiento de la red
nn = nntrain(nn, x_entrenamiento, y_entrenamiento, opts);

%% Validacion de la red

% Se introduce ruido a los datos de prueba. Densidad de 0.1
x_prueba_ruido = imnoise(x_prueba,'salt & pepper',0.1);

% Se prueba la red
[error, malos, etiquetas, salida] = nntest(nn, x_prueba_ruido, ...
                                           y_prueba);

% A consideracion del creador del toolbox, el error admitido debe ser
<0.10
assert(error < 0.10, 'Too big error');
```

Anexo B. Script para la red CNN

```
%% Cargar y acondicionamiento los datos para la red
clear,clc

% Cargar el dataset MNIST
load mnist_uint8;

% Normalizacion y adecuacion de las dimensiones de los datos
x_entrenamiento = double(reshape(train_x',28,28,60000))/255;
x_prueba = double(reshape(test_x',28,28,10000))/255;
y_entrenamiento = double(train_y');
y_datos = double(test_y');

%% Entrenamiento de la CNN

rand('state',0)

cnn.layers = {
    %capa de entrada
    struct('type', 'i')
    %capa convolucional (Capa 1)
    struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5)
    %capa subsampling (Capa 2)
    struct('type', 's', 'scale', 2)
    % capa convolucional (Capa 3)
    struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5)
    %capa subsampling (Capa 4)
    struct('type', 's', 'scale', 2)
};

% razon de aprendizaje
opts.alpha = 1;
% numero de muestras a tomar
opts.batchsize = 50;
% numero de iteraciones
opts.numepochs = 10;

% se crea la arquitectura
cnn = cnnsetup(cnn, x_entrenamiento, y_entrenamiento);
% se entrena la red
cnn = cnntrain(cnn, x_entrenamiento, y_entrenamiento, opts);

%% Validacion de la red

% Se introduce ruido a los datos de prueba. Densidad de 0.1
x_prueba_ruido = imnoise(x_prueba,'salt & pepper',0.1);

% Se realiza la validacion

% [er, bad] = cnntest(cnn, x_prueba, y_datos);
```

```
net = cnnff(cnn,x_prueba_ruido);
[~,salida] = max(net.o);
[~,etiquetas] = max(y_datos);
malos = find(salida ~= etiquetas);
error = numel(malos) / size(y_datos, 2);

% Grafica del error, en este caso, error cuadratico medio
figure; plot(cnn.rL);

% A consideracion del creador del toolbox, el error admitido debe ser
<0.12
assert(error<0.12, 'Too big error');

%% Visualizacion de las capas

% Capa 1: convolucional

tam = size(cnn.layers{2,1}.a{1,1},1);
capa1=reshape(cnn.layers{2,1}.a{1,1},tam*tam,[]);
figure,visualize(capa1);
title('Capa 1, Convolucional');

% Capa 2: subsampling

tam = size(cnn.layers{3,1}.a{1,1},1);
capa2=reshape(cnn.layers{3,1}.a{1,1},tam*tam,[]);
figure,visualize(capa2);
title('Capa 2, Subsampling');

% Capa 3: convolucional

tam = size(cnn.layers{4,1}.a{1,1},1);
capa3=reshape(cnn.layers{4,1}.a{1,1},tam*tam,[]);
figure,visualize(capa3);
title('Capa 3, Convolucional');

% Capa 4: subsampling

tam = size(cnn.layers{5,1}.a{1,1},1);
capa4=reshape(cnn.layers{5,1}.a{1,1},tam*tam,[]);
figure,visualize(capa4);
title('Capa 4, Subsampling');
```